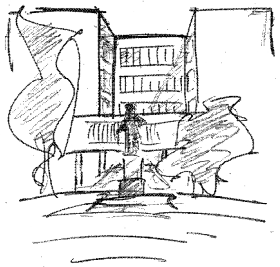
[P273]
Пројектовање база података

4

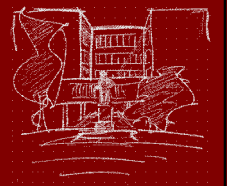


Саша Малков

Универзитет у Београду
Математички факултет

2023/2024

[P273]
Пројектовање база података



Саша Малков

Тема 6

"Модел" класа

УМЛ и моделирање база података

- Мотивација и циљ:
  - УМЛ је опште познат језик за моделирање софтвера
    - *assert(students.foreach(s: s.isUmlExpert()))*
  - Приближавање језика за моделирање података језику за моделирање информационих система и програма
- У више наврата је предлагано да се направи посебан „профил“ за моделирање података
  - Постоје незваничне радне верзије 2001-2005, али без званичних корака након тога
  - Испоставило се да није неопходан – сасвим добро је и без њега

Модел класа података

- УМЛ дијаграм класа већ описује структуру класа
- Идеја је да се исти дијаграм искористи за моделирање података:
  - боље представља семантику односа од дијаграма табела РМ
  - боље представља семантику односа чак и од правог ЕР дијаграма
  - има мање недостатке који могу да се превазиђу

Разлика у концепту модела (1)

- Модели класа и релација се концептуално разликују:
  - релације (табеле) су скупови података
    - као и скупови ентитета и односа у моделу ЕР
  - класе су типови података

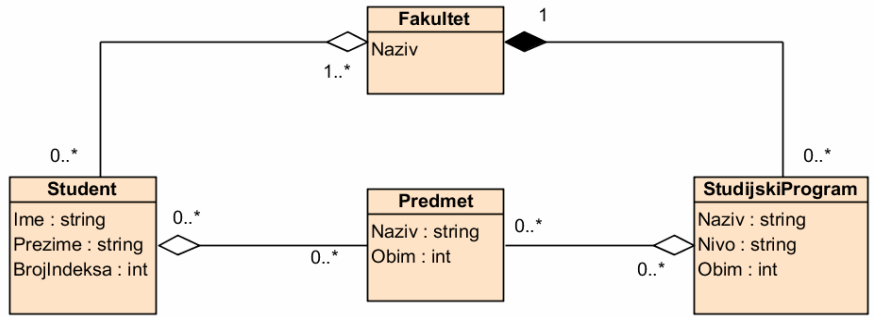
Разлика у концепту модела (2)

- Може да се превазиђе у пракси:
  - а) ако нам је потребно да моделирамо више скупова истог типа, можемо да уведемо више наследника класе која одређује тип
    - то није семантички сасвим исправно
      али се тако јасно види „сличност“ типова појединачних скупова ентитета
    - базна класа, која служи само као тип, може да се посебно означи
      - нпр. као “апстрактна”
  - б) свака “класа” може да се означи – да би се знало да ли је тип или скуп
    - типови/класе се означавају са “«*type*»”
    - скупови ентитета се означавају са “«*persistent*»”
    - то додатно оптерећује дијаграм, па се ради само када је неопходно
  - в) проблем се игнорише и сматра да је класа *скуп* а не *тип*
    - у реду је само за неке мање домене, где се не очекује више скупова истог типа

УМЛ дијаграм класа података

- За пројектовање модела базе података користи се тзв. ***концептуални дијаграм класа***
  - у УМЛ-у се обично назива ***дијаграм класа домена***
  - у раду са БП се назива и ***дијаграм класа података***
- За разлику од уобичајеног дијаграма класа:
  - у првом плану су атрибути и односи
  - понашање се скоро потпуно занемарује
  - и енкапсулација је у другом плану
- За специфичне ознаке користе се различите допуне УМЛ-а

Пример дијаграма класа података



Допуне УМЛ-а

- УМЛ садржи стандардизоване концепте који омогућавају увођење нових начина означавања:
  - стереотипови
  - означене вредности
  - проширења

Стереотипови у УМЛ-у

- Стереотип представља врсту *шаблона*:
  - апстракцију општијег случаја нечега
  - неки предефинисани скуп особина
  - неко предефинсано понашање
- Користе се за навођење додатних декларација или напомена
  - “шта конкретан елемент представља у моделу”
- Могу да се наводе и уз “класе” и уз атрибуте

Означавање стереотипова

- Подразумевано означавање стереотипова је навођењем назива између двоструких изломљених заграда:
  - <<abstract>>
  - <<persistent>>
  - <<entity>>
  - <<view>>
  - <<table>>
  - <<generated>>
  - <<auto>>
  - <<key>>
- Обично се наводи изнад, испред или испод назива класе (или објекта или атрибута)

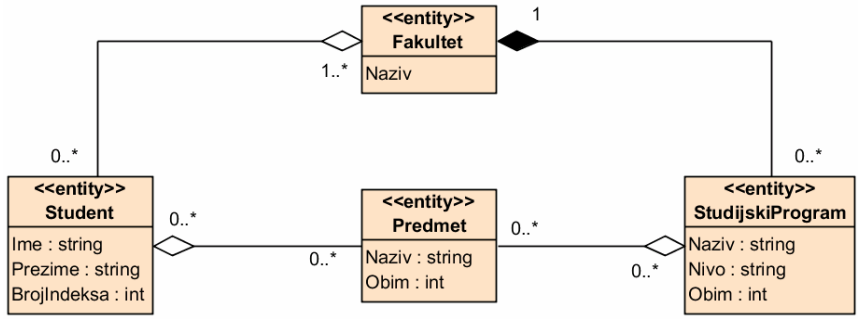
Стереотипови класа

- Уобичајени стереотипови класа у дијаграмима класа података су:
  - <<entity>>, <<persistent>>, <<set>> - концептуални или логички модел ентитета, садржај објеката се трајно записује и представља
    - <<persistent>> може да bude општија ознака да се садржај објеката трајно записује али без прејудицирања форме
    - <<set>> - никада нема понашање
  - <<type>> - ради се о типу, а не о скупу ентитета
  - <<table>> - наглашава се да је у питању табела у логичком моделу
  - <<view>> - наглашава се да је у питању поглед у логичком моделу
  - ...

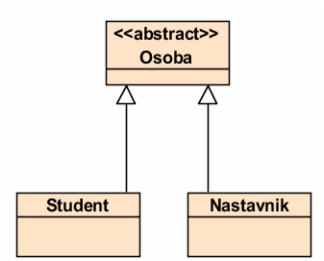
Стереотипови атрибута

- Уобичајени стереотипови атрибута су:
  - <<key>> - атрибут припада примарном кључу
    - може и <<PK>>
  - <<generated>> - атрибут је изведен, тј. израчунава се аутоматски на основу других атрибута
  - <<auto>>, <<autokey>> - аутоматски генерисан атрибут, обично сурогат кључ
  - <<unique>> - атрибут има јединствену вредност
  - <<fk>> - атрибут је референца (или део референце)
    - ретко се користи у концептуалном моделу, чешће у логичком моделу
  - <<null>> - атрибут може да има недефинисану вредност
  - <<not null>> - атрибут не сме да има недеифнисану вредност
  - ...

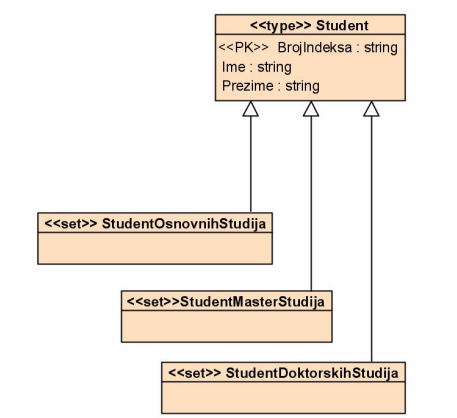
Пример дијаграма са стереотиповима



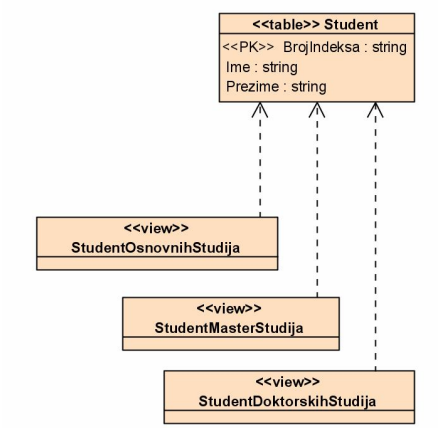
Пример дијаграма са стереотиповима (2)



Пример дијаграма са стереотиповима (3)

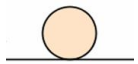


Пример дијаграма са стереотиповима (4)

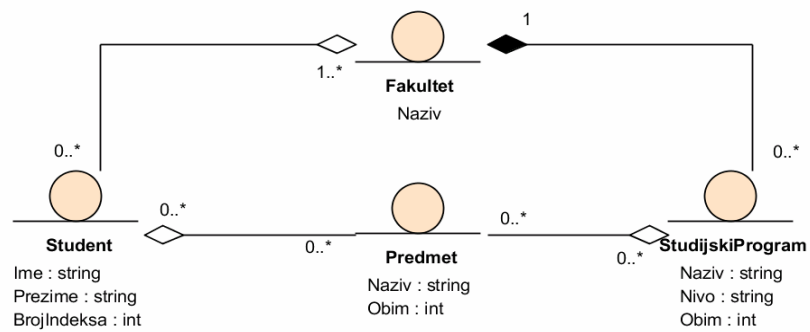


Означавање симболима

- За неке уобичајене стереотипове су уведене и графичке ознаке
- Оне обично стоје као допуна у десном горњем углу уобичајеног симбола за ту врсту објеката
- Могу чак и да замене основни графички симбол неке врсте објеката
  - ако стереотип представља значајну потклasu те врсте објеката, не пример „актер“ или „перзистентни“



Пример симбола уместо стереотипова



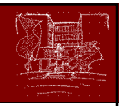
Измењено представљање “класе”

- Неки алати омогућавају да се стереотипови који одговарају скуповима подацима представљају налик на табеле у дијаграму табела – са заобљеним угловима
- Могу да се користе и боје

Означене вредности

- Уопштење стереотипова су “означене вредности”
- Наводе се као листа парова име-вредност између витичастих заграда:
  - { isPersistent= true, minCount=5 }
- У начелу, стереотип је еквивалентан означеној тачној вредности одговарајућег мета-атрибута:
  - <<abstract>> == { isAbstract=true }

Концепт проширења



- *Проширење* је елемент који се повезује са основним елементом стрелицом са попуњеним троуглом на крају
  - стрелица је усмерена од основног елемента ка проширењу
- Може да се користи за додавање особина или за описивање основног елемента
- Обучно је опционо, али може да буде и *неопходно*, што се означава са „{*required*}” изнад стрелице

Односи

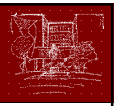
- Асоцијација
- Агрегација
- Композиција
- Наслеђивање

у основи није другачије него код уобичајених дијаграма класа

Асоцијација

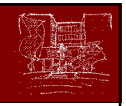
- Однос између две класе
- Може да буде:
  - *функционалан* – „уради нешто за мене“
  - *структурални* – „буди нешто за мене“
- У контексту модела података важнија је и скоро искључиво се користи ***структурална*** асоцијација
- Означава да једна класа „зна“ за неке објекте друге класе и на неки начин управља њима или комуницира са њима
  - у контексту података, “омогућава” да се комуницира са њима

Агрегација



- Посебна врста асоцијације
- Имплицира да се објекат једне класе "*састоји*" од (једног или више) објеката друге класе
- Представља слабији облик структуралне асоцијације:
  - исти део може да буде "*садржан*" у више сложених објеката
    - тј. да буде дељен
  - ако се сложени објекат обрише, делови могу да наставе да постоје
- Понекад се практично свака структурална асоцијација, чија је кардиналност "1-више", посматра као форма агрегације
  - али то није добро радити ако семантика није одговарајућа

Композиција

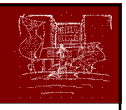


- Представља јачи облик структуралне асоцијације:
  - увек је бинарни однос
  - однос целина/део
  - један део може да припада само једном сложеном објекту
  - ако се сложени објекат обрише уобичајено понашање је да се обришу и сви делови
- Одговара концепту "*слабе ентитетске релације*" ЕР-модела
  - елементи композиције одговарају слабом ентитету

Наслеђивање

- Однос који је суштински за ОО моделирање, па тиме и за модел класа података
- Основна класа представља општији случај изведених класа
  - "*генерализација*"
- Изведене класе представљају посебне случајеве базне класе
  - "*специјализација*"
- (Релациони модел нема, а модел ентитета и односа има одговарајући концепт)

Наслеђивање (2)



- Као и у вези хијерархија у моделу ЕР, хијерархије у дијаграмима класа података често се граде на основу структуре, а не на основу понашања
- Када год из дијаграма није очигледно по ком принципу је изграђена хијерархија, то је потребно да се нагласи
- Ако се посебно наводе типови, онда је потребно да се наведу и значајне инстанце скупова одговарајућих објеката, као имплементација типа

Навигација (1)

- Модел класа података често подразумева да се за реферисање других објеката користе неки видови јединствених идентификатора, тзв. *OID*
  - то начелно одговара концепту примарног кључа

Навигација (2)

- Већина ООБП подразумева да идентификатор објекта нема никакво семантичко значење и његова вредност често чак не може ни да се види или промени
  - такви кључеви идентификују *променљиве*, а не податке
  - зато то строго формално није у складу са релационим моделом података

Навигација (3)

- Ако не прејудицирамо имплементацију, онда сурогат кључеве можемо да не наводимо
  - такав приступ је на концептуалном нивоу сасвим прихватљив, чак и пожељан
    - не оптерећујемо се увођењем нових атрибута сурогат кључа
  - прихватљив је и на логичком нивоу
    - ако се ради о објектном моделу
    - тј. све док не преведемо модел у релациони модел
    - пожељно је да се ту виде сви атрибути, па и сурогат кључеви
  - није прихватљив на физичком нивоу
    - неопходно је да се ту виде сви атрибути, па и сурогат кључеви
    - ту морамо да знамо све о свим атрибутима који се праве у БП
- Наравно, природне кључеве увек наводимо зато што представљају део модела домена

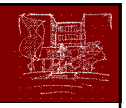
Односи у моделу података

- У моделу класа података већина односа има структуралну природу
  - због тога већина асоцијација може да прерасте у агрегацију, или чак композицију
    - али не увек!!!

Асоцијација или агрегација?

- Да ли *факултет* садржи *студенте*?
  - суштински можемо да одговоримо: "*не*"
    - студенти *нису* физички део факултета
  - али на нивоу података можемо да одговоримо: "*да*"
    - подаци о *уписаним* студентима *јесу* део података о факултету
    - штавише, ту може да се говори и о *композицији*
      - студент је *слаб* ентитет, не постоји ако није уписан на факултет
- Да ли *студент* садржи *предмете*?
  - суштински "*не*"
  - на нивоу података "*да*", али "*уписане предмете*"
- У оба случаја можемо да користимо агрегацију
  - факултет *има* скуп уписаних студената
  - студент *има* скуп уписаних предмета

Асоцијација или агрегација? (2)



- Да ли је грешка ако се користи агрегација "***факултет* садржи *студенте***"?
- Не! Али само ако се прецизира да су то *уписани* студенти.
  - Чак може и композиција – не постоји студент који није уписан на факултет
- Да ли је грешка ако се ту не користи агрегација?
- Не!
- Па шта је онда боље?
- Зависи од контекста...
  - ако правимо ИС факултета, боље је са агрегацијом
  - ако правимо нешто друго, где је информација о студијама *споредна* а не *примарна*, онда може да буде боље да користимо асоцијацију
- Основни критеријум је разумљивост
  - Концептуални модел служи за што тачније и јасније описивање домена!
    - Тежимо да представимо што више информација (агрегација је додатна инф.).
    - Шта ће неко ко чита дијаграм лакше и тачније разумети?

Разлике између нивоа моделирања

- Разлике у начину представљања модела на концептуалном / логичком / физичком нивоу сличне су као код дијаграма табела
- Концептуални дијаграм не мора да садржи (и обично не садржи):
  - додатне атрибуте сурогат кључева
  - додатне атрибуте страних кључева
  - ознаке кључева (осим можда природног кључа)
  - прецизне типове атрибута (мада је пожељно макар оквирно)
- Физички дијаграм садржи и:
  - атрибуте сурогат кључева (ако су потребне)
  - атрибуте везаних табела који чине стране кључеве
  - ознаке кључева и врста кључева
  - тачне типове атрибуте

Разлике између нивоа моделирања (2)

- ...
- Логички дијаграм у основи може и "овако" и "онако" али...
- Ако се дијаграм класа података користи само као дијаграмска техника за представљање логичког модела који није на ООМ, онда мора да садржи све оно што се очекује од логичког модела за одговарајући модел података
- На пример, ако се користи за представљање логичког модела коме одговара РМ података, онда мора да садржи све оно што очекујемо да садржи тај логички модел

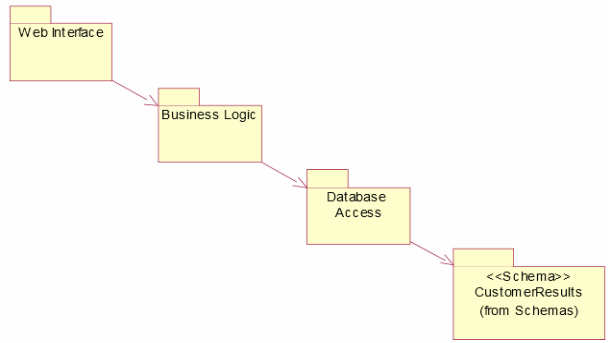
Односи на различитим нивоима моделирања

- Видели смо да исте односе често можемо да моделирамо *исправно* на различите начине
- Избор често зависи од нивоа модела:
  - на концептуалном нивоу, циљ је да очувамо семантику односа из домена проблема
  - на логичком нивоу, циљ је да остваримо стабилну и једнозначну структуру
  - на физичком нивоу, циљ је (поред претходног) да омогућимо добре перформансе

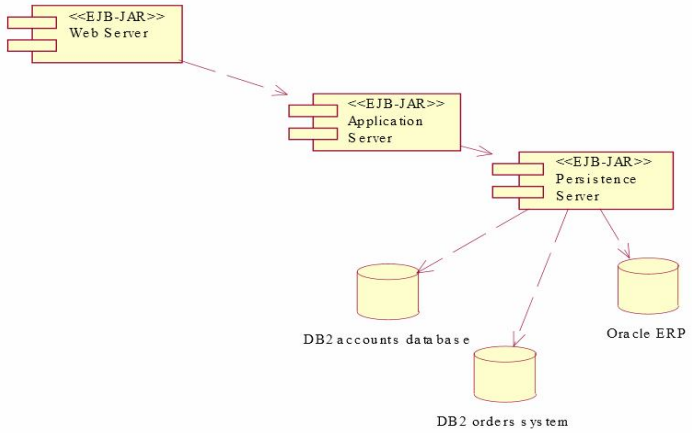
Груписање

- На концептуалном нивоу
  - табеле се групишу у схеме
  - класе се групишу у пакете
- На физичком нивоу
  - табеле се
    - групишу у просторе за табеле
    - рапоређују на чворове (сервере)
  - (функционалне) класе се
    - групишу у компоненте
    - распоређују на чворове (сервере)

Концептуално груписање



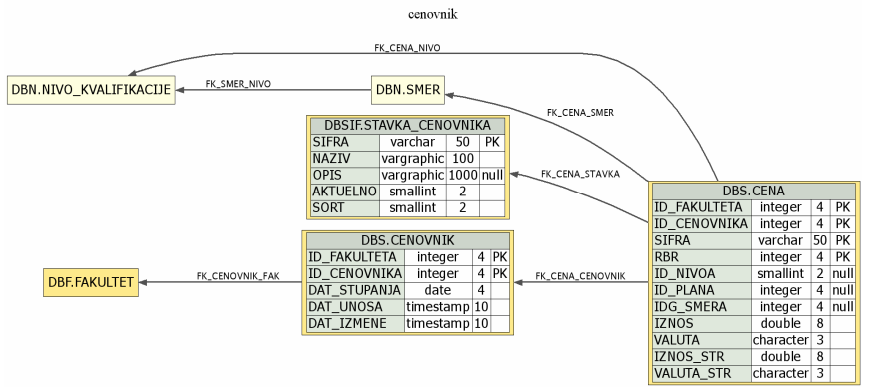
Физичко груписање



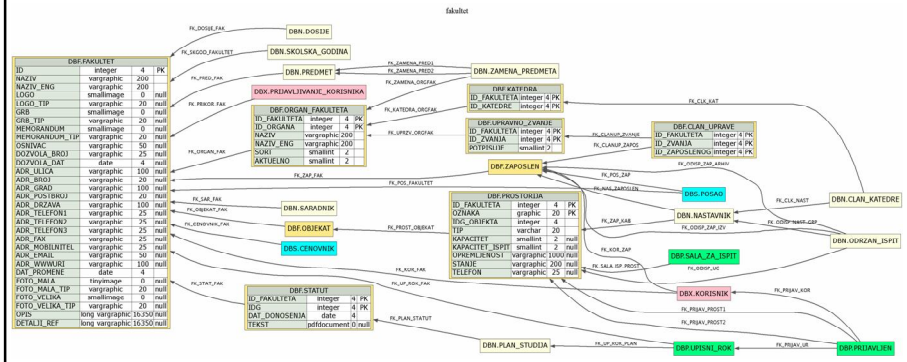
Груписање ентитета

- Иако су дијаграми класа података штедљивији у односу на заузети процес од дијаграма ЕР, ипак је често потребно да се примењује поступак груписања ентитета
- Слично као код ЕР
  - скривамо атрибуте
  - скривамо посредно повезане елементе
  - користимо различите боје за означавање
  - можемо да групу представимо пакетом и да уз коришћење пакета наводимо назив класе која се користи

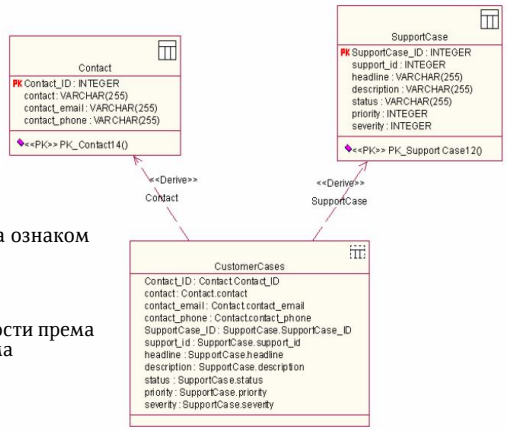
Груписање ентитета



Груписање ентитета

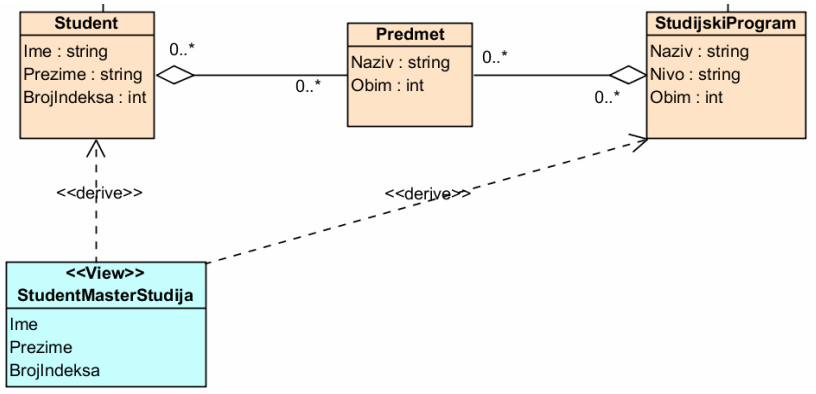


Представљање погледа



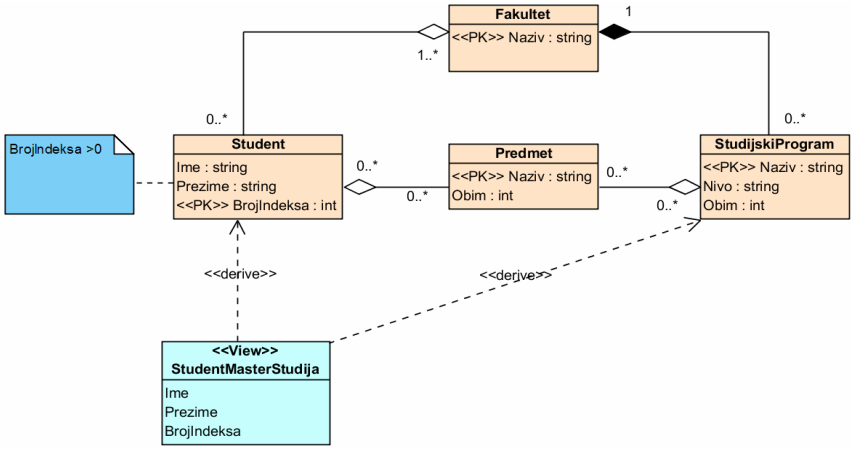
- Погледи се представљају са ознаком стереотипа у углу
  - може и симбол
  - или друга боја
  - уводи се и однос зависности према базним класама/табелама

Представљање погледа (2)



Означавање кључева

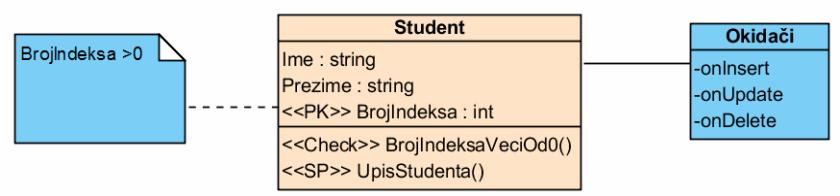
- Симболима или текстом



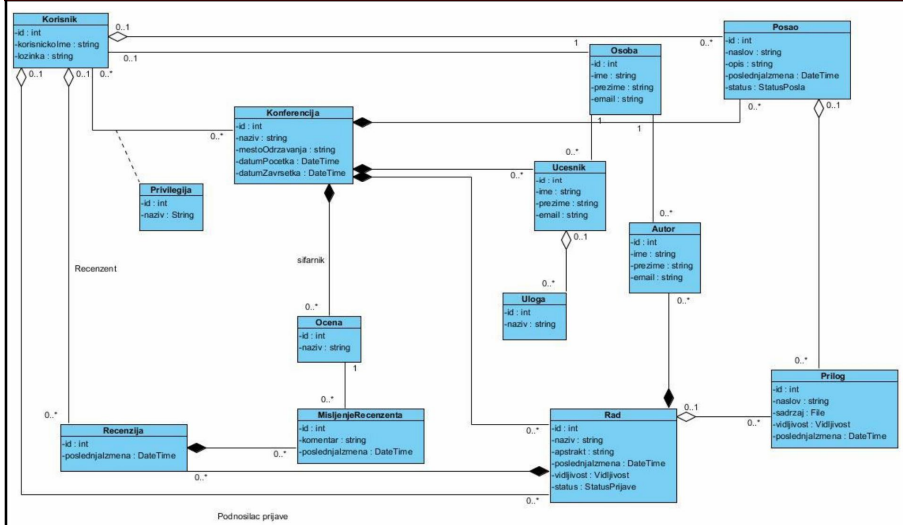
Услови интегритета, индекси,...

- Разне друге ствари могу да се наводе у оквиру класа у одељку понашања
  - али неки алати не пружају такву слободу
- Алтернативно се наводе као повезане “класе” или коментари

- услови интегритета
- индекси
- окидачи
- серверске процедуре
- ...



Пример 1 –Систем за подршку организовања научних скупова



Литература за тему

- Molina, **Database Systems**, 2.ed, *Pearson Prentice Hall* (2009)
- Teorey et al., **Database Design**, *Morgan Kaufmann* (2009)
- Muller, **Database Design for Smarties – Using UML for Data modeling**, *Academinc Press* (1999)
- Alvaro Monge, **Database design with UML and SQL**, 4.ed.
  - https://web.csulb.edu/colleges/coe/cecs/dbdesign/dbdesign.php