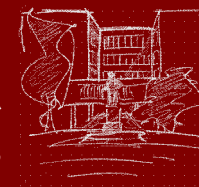


[P273] Пројектовање база података 4.4



Саша Малков
Универзитет у Београду
Математички факултет
2023/2024

[P273] Пројектовање база података Саша Малков



Тема 7.3

Релациони модел података (интегритетни део)

Интегритет и конзистентност



- *Конзистентност* базе података је *сагласност* садржаја базе података са стварним простором који се базом података моделира
 - представља *тачност* или *исправност* садржаја базе података
 - база података није конзистентна ако садржи нетачне или непотпуне податке
- *Интегритет* базе података подразумева *логичку исправност* садржаја базе података
 - логичка исправност се остварује проверавањем испуњености логичких правила интегритета
- У пракси провера интегритета представља део провере конзистентности који може да се формализује / аутоматизује

Интегритет и конзистентност (2)



- Конзистентност базе података се остварује на два основна начина:
 - аутоматским старањем о интегритету
 - СУБП аутоматски проверава интегритет базе података проверавањем испуњености дефинисаних правила интегритета
 - имплементацијом трансакционих механизма
 - СУБП се стара да свака јединица промене (трансакција) буде или у потпуности изведена или да не остави никакав траг (у случају прекида)
- Наравно, конзистентност у великој мери зависи и од исправности конкретних података и конкретних изведених јединица промена
 - тај део оговорности не може да се повери СУБП-у



Интегритетни део модела

- Теоријски модел базе података може да буде од велике помоћи при дефинисању и остваривању правила интегритета
- *Интегритетни део модела* чине концепти и механизми који омогућавају да се аутоматизује проверавање задовољености одређених услова
- Релациони модел је у томе посебно добар



Интегритетни део релационог модела

- База података (тј. њена вредност) се описује релацијама
- Услови интегритета у релационом моделу се описују предикатима (истинитосним формулама) над релацијом или базом података
- Формалност структурног дела модела олакшава формулисање услова интегритета



Врсте услова интегритета

- Општи услови интегритета (имплицитни)
 - подразумевају се на нивоу модела и имплементације СУБП
 - одговарајућа логичка правила су уграђена у модел и имплементацију
 - не дефинишу се експлицитно за сваку релацију или атрибут
 - при одређивању структуре базе података се одређују и конкретне специфичне инстанце (тј. параметри примене) тих услова
- Специфични услови интегритета (експлицитни)
 - они који се односе на поједину релацију, атрибут или базу података
 - одређују се при одређивању структуре базе података
 - експлицитно се наводе одговарајући логички услови



Општи услови интегритета

- Најважније врсте логичких правила интегритета
 - Интегритет домена
 - правило: свака вредност неког атрибута у бази података мора припадати одговарајућем домену атрибута
 - Интегритет (примарног, јединственог) кључа
 - правило: јединственост торки у релацији се остварује дефинисањем примарног кључа, чија вредност једнозначно одређује торку релације
 - Интегритет страног кључа
 - правило: интегритет односа међу релацијама се остварује дефинисањем страног кључа, чија вредност једнозначно одређује торку повезане релације
- За сваки конкретан случај (атрибут, релација, однос) се дефинишу конкретне појединости



Интегритет домена

- “Свака релација мора да има тачно одређен домен”
 - Свака вредност неког атрибута у бази података мора припадати одговарајућем домену атрибута
 - Последица структурног дела модела:
 - атрибут је функција која слика ентитет у задати домен атрибута
- Технички се остварује одређивањем домена при дефинисању атрибута у оквиру наредбе за прављење табеле базе података
 - Домен је неки од подржаних типова базе података и може да обухвата
 - дужину податка
 - опциону декларацију јединствености
 - опциону декларацију подразумеване вредности
 - (већина имплементација омогућава и декларисање да ли домен обухвата и тзв. недефинисане вредности)
 - Неке имплементације омогућавају и кориснички дефинисане типове - *UDF*



Интегритет домена (2)

- У случају РСУБП DB2 интегритет домена се имплементира одређивањем типова колона при прављењу табела:

```
CREATE TABLE <име табеле> (
  ...
  <име колоне> <тип колоне>... ,
  ...
)...
```



Појам кључа

- Подсетимо се:
 - **Најкључ** K релације R је подскуп њених атрибута $K \subseteq A(R)$ за који важи:
 - скуп атрибута K једнозначно одређује торке релације
 - **Кључ** (или **кључ-кандидат**) релације R је наткључ K за који важи:
 - не постоји прави подскуп скупа K који је наткључ релације R
- Приметимо:
 - Свака релација има најмање кључ, а може да их има више
 - Сваки наткључ садржи бар један кључ
- Последице:
 - За сваке две торке x и y релације R , важи $K(x) \neq K(y)$
 - Пројекција релације R на атрибуте кључа K има исти број елемената као и R



Врсте кључева

- Уводимо још неке важне појмове у вези са кључевима:
 - **Примарни кључ** – један изабран кључ се проглашава за примарни
 - СУБП проверава јединственост торки релације проверавањем јединствености вредности примарног кључа
 - **Алтернативни кључ** или **јединствени кључ** – сваки кључ осим примарног
 - **Страни кључ** – скуп колона који представља кључ повезане релације
 - вредност страног кључа једнозначно одређује торку повезане релације
 - **Сложени кључ** – кључ који се састоји од више од једног атрибута
 - **Обједињени кључ** – сложени кључ који се састоји од два или више страних кључева (и само од њих)
 - **Сувођај кључ** – вештачки уведен атрибут са улогом кључа
 - уводи се када релација нема природан кључ или када је природан кључ неприкладан за употребу (има много атрибута или је незгодан тип)



Интегритет примарног кључа

- “Свака релација има бар један кључ”
 - Последица је структурног дела модела:
 - релација је скуп, па мора да садржи јединствене вредности
- Технички се остварује дефинисањем тачно једног *примарног кључа* при прављењу табеле
 - За примарни кључ се проглашава један од кључева релације
 - за који се очекује да ће најчешће бити употребљаван за приступање подацима
 - чији домен не садрже недефинисане вредности
 - Примарни кључ може да се састоји од једне или више колона
 - СУБП користи примарни кључ за проверавање јединствености торки релације



Интегритет примарног кључа (2)

- У случају РСУБП DB2 интегритет примарног кључа се имплементира одређивањем примарног кључа при прављењу табеле:

```
CREATE TABLE <име табеле> (
  ...
  [CONSTRAINT <име кључа>]
  PRIMARY KEY ( <листа имена колона> ) ...
  ...
  )...
```



Интегритет јединствености

- Интегритету примарног кључа се придружује и *интегритет јединствености* (назива се и *интегритет јединствености*):
 - “Примарни кључ не сме да садржи недефинисане вредности, ниши две торке једне релације смеју имати исте вредности примарних кључева”
- У доследним имплементацијама то је исто као интегритет примарног кључа
 - примарни кључ по дефиницији не сме да садржи недефинисане вредности
 - јединственост вредности примарног кључа је имплицирана јединственошћу торки у релацији
- Ипак, у недоследним имплементацијама:
 - интегритет јединствености је јачи од интегритета кључа
 - зато што постоји подршка за недефинисане вредности
 - могу да постоје две идентичне торке у релацији



Интегритет јединственог кључа

- Поред примарног кључа, могу да се експлицитно декларишу и *јединствени кључеви*
 - имају исте карактеристике као примарни кључеви
 - начелно *исте* а у пракси могу да буду *сличне*
 - очекује се да се користе при реферисању торки ређе од примарног кључа
- Основна намена јединственог кључа је имплементација додатних услова интегритета јединствености торки
 - ако релација има више кључева, један се проглашава за примарни а остали (алтернативни кључеви) могу да се дефинишу као јединствени



Интегритет јединственог кључа (2)

- У случају РСУБП DB2 интегритет јединственог кључа се имплементира дефинисањем јединственог кључа при прављењу табеле:

```
CREATE TABLE <име табеле> (
  ...
  [CONSTRAINT <име кључа>]
  UNIQUE ( <листа имена колона> ) ...
  ...
  )...
```



Интегритет јединственог кључа (3)

- У неким случајевима може да буде потребно да се проверава јединственост кључева који могу да имају недефинисане вредности
 - тј. не испуњава услов интегритета јединствености
 - на пример, ако за неки атрибут X није могуће одредити вредности за све торке, али оне које су познате морају да буду јединствене
- Доследне имплементације то не омогућавају
 - али неке нуде заобилазни пут
 - на пример, на DB2 може да се направи индекс који уз индексирање обавља и проверу јединствености дефинисаних вредности:
 - CREATE UNIQUE INDEX ... EXCLUDE NULL KEYS ...



Референцијални интегритет

- Референцијални интегритет представља услове о међусобним односима које морају да задовољавају торке двеју релација:
 - не сме се обрисати торка на коју се односи нека торка неке релације у бази података, нијих се сме тако изменити да референца постане неисправна
 - не сме се додати торка са неискравном референцом (иаковом да не постоји торка на коју се односи)
- У релационом моделу се референцијални интегритет остварује путем *интегришета страни кључа*



Референцијални интегритет (2)

- У случају недефинисане вредности, услови референцијалног интегритета су измењени:
 - не сме се обрисати торка на коју се односи нека торка неке релације у бази података, нијих се сме тако изменити да референца постане неискравна
 - не сме се додати торка са неискравном референцом (иаковом да не постоји торка на коју се односи)
 - референца која садржи недефинисане вредности је исправна (и не референцира ништа) ако је у постојности недефинисана (тј. сви њени атрибути су недефинисани)
- У већини имплементација треће правило је додатно релаксирано:
 - референца која садржи недефинисане вредности је исправна (и не референцира ништа) ако је макар један њен део недефинисан



Интегритет страног кључа

- Скуп FK атрибута релације R је њен *страни кључ* који се односи на базу релацију B ако важе следећи услови:
 - релација B има примарни кључ PK
 - домен кључа FK је идентичан домену кључа PK
 - свака вредност страног кључа FK у торкама релације R је идентична кључу PK бар једне торке релације B
 - (интегритет кључа имплицира јединственост)
- За релацију R се каже да је *зависна* од базне релације B
- Базна релација B се назива и родитељском релацијом



Интегритет страног кључа (2)

- У случају недефинисане вредности, скуп услова је измењен:
 - релација B има примарни кључ PK
 - домен кључа FK је идентичан домену кључа PK , или је проширен недефинисаним вредностима
 - свака вредност кључа FK у торкама релације R је или недефинисана или је идентична кључу PK бар једне торке релације B
 - вредност страног кључа FK у торкама релације R је недефинисана ако све вредности атрибута страног кључа имају недефинисану вредност
- У већини имплементација је четврто правило додатно релаксирано:
 - вредност страног кључа FK у торкама релације R је недефинисана ако бар један атрибут страног кључа има недефинисану вредност



Интегритет страног кључа (3)

- Поштовање интегритета страног кључа при мењању садржаја базе података се одређује:
 - правилом брисања (бира се тачно једно) и
 - правилом ажурирања (бира се тачно једно)



Правило брисања

- Правила брисања (бира се тачно једно): “У случају покушаја брисање торке базне релације B , за коју постоји зависна торка релације R (она чији је страни кључ једнак примарном кључу торке која се покушава обрисати) онда...”
 - активна забрана брисања – *RESTRICT*
 - “...забрањује се брисање торке из релације B .”
 - пасивна забрана брисања – *NO ACTION*
 - слично као активна забрана, али се одлаже провера до самог краја брисања, зато што можда постоји линија каскадних правила која ће обрисати зависну торку
 - каскадно брисање – *CASCADE*
 - “...бришу се све одговарајуће зависне торке R .”
 - постављање недефинисаних вредности – *SET NULL*
 - “...у свим зависним торкама релације R атрибути страног кључа се постављају на недефинисане вредности.”
 - постављање подразумеваних вредности – *SET DEFAULT*
 - “...у свим зависним торкама релације R атрибути страног кључа се постављају на подразумеване вредности.”



Правило ажурирања

- Правила ажурирања (бира се тачно једно): “Ако се покуша мењање примарног кључа торке релације *B*, за коју постоји зависна торка релације *R* (она чији је страни кључ једнак примарном кључу торке која се покушава обрисати) онда...”
 - активна забрана мењања – *RESTRICT*
 - “...забрањује се мењање торке релације *B*.”
 - пасивна забрана мењања – *NO ACTION*
 - слично као активна забрана, али се одлаже провера до самог краја мењања, зато што можда постоји линија каскадних правила која ће променити зависну торку;
 - каскадно мењање – *CASCADE*
 - “...на исти начин се мењају атрибути страног кључа свим зависним торкама релације *R*.”
 - постављање недефинисаних вредности – *SET NULL*
 - “...у свим зависним торкама релације *R* атрибути страног кључа се постављају на недефинисане вредности.”
 - постављање подразумеваних вредности – *SET DEFAULT*
 - “...у свим зависним торкама релације *R* атрибути страног кључа се постављају на подразумеване вредности.”



Интегритет страног кључа (4)

- У случају система DB2 страни кључ се дефинише при прављењу табеле:

```
CREATE TABLE <име табеле> (
  ...
  [CONSTRAINT <име кључа>]
  FOREIGN KEY ( <листа имена колона> )
  REFERENCES <име табеле> [ ( <листа имена колона> ) ]
  [ON DELETE <правило брисања>]
  [ON UPDATE <правило ажурирања>]
  ...
  )...
```

- Правило брисања може да буде:
 - RESTRICT, NO ACTION, CASCADE, SET NULL
- Правило ажурирања може да буде:
 - RESTRICT, NO ACTION
- У оба случаја подразумевано правило је NO ACTION



Специфични услови интегритета

- Специфични услови интегритета се дефинишу у облику логичких правила која увек морају да буду испуњена
- У односу на опсег података који се проверавају, услови могу да буду дефинисани на:
 - атрибуту
 - торки
 - релацији
 - бази података



Услов интегритета на атрибуту

- Услов интегритета на атрибуту
 - локалног је карактера, односи се на вредност једног атрибута једне торке
 - услов може да зависи само од вредности атрибута
 - може да се користи за додатно сужавање домена
 - на пример, ако целобројни атрибут мора да буде у задатом опсегу
 - може да се користи за проверу исправности сложених типова података
 - на пример, ако текстуални податак мора да има неку задату форму



Услов интегритета на атрибуту (2)

- Дефинише се при описивању атрибута табеле:

```
CREATE TABLE <име табеле> (
  ...
  <име атрибута> <тип атрибута>
    CHECK ( <логички услов> )
  ...
) ...
```

- Наведени логички услов
 - сме да реферише само тај један атрибут за који се дефинише
 - мора да буде испуњен за сваки ред табеле
 - у случају недефинисаних вредности услов мора да буде **не-лажан**
- Подржано у већини имплементација СУБП



Услов интегритета на атрибуту (3)

- Пример:

```
CREATE TABLE ... (
  GODINA INT CHECK ( GODINA BETWEEN 1980 AND 2100 ),
  ...)
```



Услов интегритета на торки

- *Услов интeгритeтa нa торкe*
 - локалног карактера, односи се на вредност једне торке
 - услов може да зависи од вредности свих атрибута торке
 - користи се за проверу исправности сложенијих сагласности атрибута у оквиру једне торке
 - на пример, ако се наводи неки интервал, може да се проверава да ли је почетна вредност интервала мања од крајње вредности интервала



Услов интегритета на торки (2)

- Дефинише се при прављењу табеле:

```
CREATE TABLE <име табеле> (
  ...
  CONSTRAINT [<име услова>] CHECK ( <логички услов> )
  ...
) ...
```

- Наведени логички услов
 - сме да реферише све атрибуте те релације
 - мора да буде испуњен за сваки ред табеле
 - у случају недефинисаних вредности услов мора да буде **не-лажан**
- Подржано у већини имплементација СУБП

Услов интегритета на торки (3)



- Пример:

```
CREATE TABLE ... (
  ...
  GOD_OD INT,
  GOD_DO INT,
  CONSTRAINT OPSEG CHECK ( GOD_OD <= GOD_DO )
  ...
) ...
```

Услов интегритета на релацији



- Услов интегритета релације

- глобалног карактера, може да се односи на све торке једне релације
- услов може да зависи од вредности свих атрибута свих торки
- користи се за проверу исправности сложенијих сагласности вредности у оквиру једне релације
 - на пример, може да се проверава да ли број торки или сума по неком атрибуту задовољавају одређене услове

Услов интегритета на релацији (2)



- Дефинише се при прављењу табеле:

```
CREATE TABLE <име табеле> (
  ...
  CONSTRAINT [<име услова>] CHECK ( <логички израз са подупитима> )
  ...
) ...
```

- Алтернативна синтакса је иста као за услове интегритета на бази података
- Обично су имплементирани слично као и услови интегритета на торкама
 - логички израз почива на подупитима над само том релацијом
 - мора да буде испуњен за табелу као целину
 - у случају недефинисаних вредности услов мора да буде **не-лажан**
- Не подржава их већина имплементација

Услов интегритета на релацији (3)



- Пример:

```
CREATE TABLE T1 (
  ...
  A INT,
  B INT,
  CONSTRAINT C1 CHECK (
    (select sum(A) from t1)
    <
    (select sum(B) from t1)
  )
  ...
) ...
```



Услов интегритета на бази података

- *Услови интегритета на бази података*
 - глобалног су карактера, односе се на садржај различитих релација
 - користе се за проверу сложенијих услова интегритета
 - на пример, ако садржај једне релације мора да буде у складу са садржајем друге релације, али на начин који превазилази могућности обичног референцијалног интегритета



Услов интегритета на бази података (2)

- Дефинише се после прављења табела на које се односи:


```
CREATE ASSERTION <име> CHECK ( <логички израз са подупитима> )
```
- При томе:
 - логички израз почива на подупитима над било којим дефинисаним релацијама
 - мора да буде испуњен за базу података као целину
 - у случају недефинисаних вредности услов мора да буде **не-лажан**
- Не подржава их већина имплементација



Услов интегритета на бази података (3)

- Пример:


```
CREATE ASSERTION ASRT_1 CHECK (
  (SELECT COUNT(*) FROM T1)
  <
  (SELECT COUNT(*) FROM T2)
)
```



Када се проверавају услови?

- Услови се проверавају при свакој промени садржаја базе података
- Некада је потребно да се услови проверавају тек на крају трансакције
- Већина РСУБП то омогућава на неки начин
 - може да се мења подразумевани начин рада
 - може да се мења начин рада у текућој трансакцији
- Наредна наредба (*DB2, Oracle, PostgreSQL,...*), одлаже све провере у текућој трансакцији одлажу до њеног завршетка:


```
SET CONSTRAINTS ALL DEFERRED
```

 - *постоји већи број опција, могу се бирати табеле и услови који се одлажу

Активно одржавање интегритета



- *Активно* одржавање интегритета подразумева да се интегритет одржава тако што се на одређене промене реагује покретањем експлицитно дефинисаног програмског кода
 - тај програмски код прави неопходна усклађивања повезаних података
- Термин *активно* потиче од извршавања експлицитно задатих програма
- Механизам активног одржавања интегритета су *окидачи* (енгл. *trigger*)

Окидачи



- Окидачи подразумевају да се на одређене промене реагује покретањем одговарајућег програмског кода
- Сваки окидач је дефинисан
 - релацијом на којој се прате промене
 - врстом промене на коју се реагује
 - програмским кодом који се извршава
 - тренутком извршавања – пре или после промене
 - грануларношћу извршавања – да ли се извршава по једанпут за сваку промењену торку или по једанпут за целу наредбу мењања
 - подацима који се реферишу у програмском коду

Окидачи (2)



- Врста промене може да буде
 - INSERT – додавање торки релацији
 - UPDATE – мењање торки релације
 - DELETE – брисање торки релације
- Тренутак извршавања може бити
 - BEFORE – пре промене
 - AFTER – после промене
- Реферисање торки може бити
 - OLD – стара вредност пре промене (не може за INSERT)
 - NEW – нова вредност после промене (не може за DELETE)
- Грануларност може да буде
 - FOR EACH ROW – извршава се за сваки промењен ред
 - FOR EACH STATEMENT – извршава се за сваку извршену наредбу

Окидачи (3)



- Програмски код који је везан за окидач се дефинише на различите начине
- У случају система DB2 то може да буде:
 - једна наредба SQL-а
 - блок наредби SQL-а
 - серверска процедура
 - и друго
- Блок наредби SQL-а почиње са BEGIN и завршава се са END

Дефинисање окидача



- Синтакса наредбе (поједностављена):

```
CREATE [OR REPLACE] TRIGGER <име окидача>
{ [NO CASCADE] BEFORE | AFTER }
{ INSERT | UPDATE | DELETE }
ON <име табеле>
[ REFERENCING {OLD | NEW | OLD TABLE | NEW TABLE } AS <име>]*
FOR EACH { ROW | STATEMENT }
[ WHEN <услов> ]
<програмски код>
```

Пример окидача



- Пример окидача:
 - аутоматски израчунава основну плату новог службеника на основу нивоа образовања:

```
CREATE OR REPLACE TRIGGER insert_set_salary
NO CASCADE BEFORE INSERT ON employee
REFERENCING NEW AS N FOR EACH ROW
BEGIN
  SET n.salary = CASE n.edlevel WHEN 18 THEN 50000
                        WHEN 16 THEN 40000
                        ELSE 25000
END
```

Пример окидача



- Пример окидача:
 - ако би у одељењима постојао број радника у одељењу, он би могао да се ажурира окидачима при додавању и брисању радника:

```
CREATE OR REPLACE TRIGGER insert_radnik
AFTER INSERT ON radnik
REFERENCING NEW AS N FOR EACH ROW
BEGIN
  UPDATE org_jedinica
  SET broj_radnika = broj_radnika + 1
  WHERE org_jed_id = n.org_jed_id
END
```

```
CREATE OR REPLACE TRIGGER delete_radnik
BEFORE DELETE ON radnik
REFERENCING OLD AS R FOR EACH ROW
BEGIN
  UPDATE org_jedinica
  SET broj_radnika = broj_radnika - 1
  WHERE org_jed_id = r.org_jed_id
END
```

Када користимо активно одржавање интегритета?



- Ради одржавања исправних вредности повезаних података у бази података
- Ради проверавања интегритета
- Ради извођења акција ван базе података



Окидачи и одржавање података у бази

- Одржавање исправних вредности података у бази података помоћу окидача је најчешће *лоша идеја*
 - то је сигнал да у бази података постоје редундантни подаци, што је потенцијално озбиљан проблем
 - редундантности ћемо посветити пажњу на наредним часовима
- Обично се оправдава перформансама
 - на пример, да не бисмо стално изнова израчунавали просечну оцену, можемо да је додамо као колону и одржавамо окидачима
 - али тада постоје и другачија, потенцијално боља решења
 - нпр. материјализовани погледи



Окидачи и проверавање интегритета

- У одсуству могућности дефинисања специфичних правила интегритета на нивоу табеле или базе података, могу да се користе окидачи
- Идеја је да се при некој измени проверава да ли је све у реду и пријављује се грешка (и прекида трансакција) ако није:
 - ... SIGNAL SQLSTATE 'ER001'
SET MESSAGE_TEXT = 'Neispravno menjanje!'
- Користи се, али најчешће *није добра идеја*
 - у иоле сложенијим трансакцијама, када више наредби успоставља исправан однос, окидач може да реагује већ на прву, пре времена



Окидачи и акције ван базе података

- Употреба окидача ради извођења акција ван базе података може да буде добар начин за повезивање базе података са другим елементима информационог система
 - на пример, када стање производа у складишту падне испод одређеног нивоа, окидач може да пошаље имејл за наручивање производа од добављача
 - или, ако је потребно ажурирати садржај неке друге базе података



Окидачи на погледима

- Савремени СУБП нуде окидаче на погледима
 - Извршавају се *уместо* наредбе за додавање, мењање или брисање торки
 - Омогућавају преусмеравање измена на релације на којима почива поглед, али и даље од тога, на сасвим друге релације
- Пример окидача на погледу:
 - Ако поглед V1 почива на релацијама T1 и T2, окидачем се решава како се мењају табеле "кроз поглед"
 - CREATE TRIGGER V1_UPDATE
INSTEAD OF UPDATE ON V1
REFERENCING NEW AS n OLD AS o
FOR EACH ROW MODE DB2SQL
UPDATE T1
SET (c1, c2) = (n.c1, n.c2)
WHERE c1 = o.c1 AND c2 = o.c2



Окидачи на погледима (2)

- Окидачи на погледима омогућавају скривање веома сложених операција којима се спољашња схема раздваја од концептуалне, или концептуална од интерне
- Пример окидача на погледу:
 - имплементира скривено имплицитно енкриптовање лозинки при записивању у базу података:
 - ```
CREATE TRIGGER UPDATE_LOGINS INSTEAD OF UPDATE
ON LOGINS REFERENCING OLD AS o NEW AS n
FOR EACH ROW MODE DB2SQL
UPDATE USERS U
SET system = n.system,
 login = n.login,
 password = encrypt(n.password)
WHERE system = o.system
AND login = o.login
AND U.user = USER
```



## Окидачи на погледима и раздвајање нивоа модела

- Основно средство раздвајања нивоа схема у релационом моделу су погледи
  - На нижем нивоу (интерна или концептуална схема) релације су нормализоване ради стабилности и нередундантности
  - На вишем нивоу (концептуална или спољашња схема) релације могу да буду денормализоване, тј. спајају се ради лакшег постављања упита и приступања подацима
  - Привилегије на погледима могу да се потпуно разликују од привилегија на релацијама
- Основно ограничење погледа:
  - Погледи над више релација не могу да се ажурирају
  - То значајно ограничава примену погледа у раздвајању нивоа модела
- Окидачи на погледима потпуно превазилазе то ограничење:
  - Омогућавају потпуну контролу над употребом погледа за раздвајање нивоа модела
  - Употпуњују и заокружују релациони модел



## Резиме релационог модела

- Пуна математичка заснованост
  - непротивречан и недвосмислен модел
- Јединствен и потпун концепт релације
  - релација моделира и ентитете и односе
  - тј. сав садржај базе података
- Једноставан концепт интегритетног дела модела
  - строга формалност модела омогућава формалност и прецизност интегритетног дела модела
- Универзалан приступ на свим нивоима
  - исти модел може да се користи на свим нивоима од интерне, преко концептуалне до спољашње схеме
  - технике релационог модела могу да се користе за потпуно раздвајање различитих нивоа схема

## Литература за тему

- Гордана Павловић-Лажетић, **Увод у релационе базе података, 2. изг. Математички факултет, 1999.**
  - доступно онлајн: <http://poincare.matf.bg.ac.rs/~gordana/urbp-2016.htm>
- Ramakrishnan, Gehrke, **Database Management Systems, 2.ed, 2000.**
- Codd, **A relational model of data for large shared data banks, Comm.ACM, 13(6), 1970.**
- Codd, **Extending the database relational model to capture more meaning, ACM ToDS, 4(4), 1979.**
- Codd **The Relational Model for Database Management – Version 2, AddisonWesley Publ. Inc., 1990.**
- Darwen, Date, **The Third Manifesto, 1995.**
- IBM, **Database Administration Concepts and Configuration Reference, 2012.**