OSNOVE BAZA PODATAKA

MODEL PODATAKA

BAZE PODATAKA

- U osnovi svakog sistema za automatizovanu obradu podataka
  - od najjednostavnijih
  - do Integralnih Informacionih Sistema
- je BAZA PODATAKA (BP).

U tom smislu se BP ne može posmatrati van konteksta nekog sistema za automatizovanu obradu podataka – danas – Informacionog Sistema (IS).
Da se kratko osvrnemo na pojama IS – (Šta je IS ?)

INFORMACION SISTEM

Informacion sistem je model realnog sistema u kojem deluje.

Postupak projektovanja IS svodi se na modelovanje posmatranog realnog sistema.

Metodologija projektovanja IS obuhvata:

- Modelovanje Procesa
- Modelovanje Podataka
- Modelovanje resursa

ISTORIJSKI RAZVOJ (IS)

ISTORJSKI: Automatizovana obrada (posmatrana u odnosu na podatke) ima dva perioda ili dva pristupa:

- **Klasična obrada podataka, i**
- **Koncepcija baza podataka**

Klasična obrada podataka

- Podaci su uglavnom smešteni u skup nezavisnih datoteka
- Model podataka (struktura datoteka) nastajala na osnovu “zahteva” programa
- Isti podaci se retko koriste u više aplikacija
- Izmene zbog potrebe jednog programa utiču na menjanje svih drugih programa

Klasična obrada podataka

Nedostaci:

- Nepovezanost aplikacija
- Redudantnost podataka
- Čvrsta veza između programa i podataka

Prednost:

- Jednostavnost rešenja

Koncepcija baza podataka

U rešavanju pomenutih problema pošlo se ka integrisanju podataka i nastaje (početak 60-tih godian):

- **Koncepcija baza podataka**

Osnovni ciljevi koncepcije baze podataka su:

- Podatke IS integrisati u jednu fizičku strukturu podataka. Dolazi se do koncepta ŠEME BAZE PODATAKA
- Sve obrade koriste standardizovane softverske rutine SISTEM ZA UPRAVLJANJE BAZOM PODATAKA (SUBP)

Koncepcija baza podataka

Za raliku od klasične obrade podataka u koncepciji baza podataka:

- Model podataka nastaje nezavisno od programa kao model realnog sistema (na osnovu osobina objekata i njihovih veza u realnom sistemu).
  - Takav model se naziva Konceptualni model podataka.
- Konceptualni model podataka je nezavisan od tekućih obrada – pogodan za sve, pa i za nepredviđene buduće obrade.

INFORMACIONI SISTEM
- MODEL PODATAKA -

INFORMACIONI SISTEM - je model realnog sistema u kome deluje, pa se postupak projektovanja Informacionog sistema svodi na neku vrstu modelovanja realnog sistema u kojem deluje.

Metodologija projektovanja informacionih sistema uključuje:

- Modeliranje podataka,
- Modeliranje procesa, i
- Modeliranje resursa

METODE ZA PROJEKTOVANJE

- Za modeliranje podataka i procesa postoje standardizovane metode, na primer
- SSA – za modelovanje procesa i
- MOV – za modelovanje podataka;
- dok za modelovanje resursa ne postoje standardizovane i stroge metode.

MODEL PODATAKA

- Mi ćemo se u okviru predmeta baviti delom IS koji se odnosi na podatke.
- MODEL PODATAKA – je skup međusobno povezanih podataka koji opisuju objekte, njihove veze i osobine realnog sistema.
- U modelu podataka ne opisuje se potpuni skup znanja o sistemu, već se vrši odabir i opis relevantnih karakteristika sistema.

Model podataka se sastoji od tri komponente:

- Strukturne,
- Integritetne, i
- Operacijske

KOMPONENTE MODELA PODATAKA

- **Strukturnu komponentu** – čini skup primitivnih koncepata i skup pravila za izgradnju složenih koncepata.
- **Integritetnu komponentu** – čini skup uslova integriteta koji se iskazuju preko: dozvoljenih podataka u jednom tipu objekta, dozvoljenih vrednosti podataka nekog obeležja, dozvoljenih veza između tipova objekata isl.
- **Operacijsku komponentu** – modela podataka – čini skup koncepata koji omogućuju interpretaciju dinamičkih karakteristika skupa podataka i promenu stanja podataka u bazi podataka u skladu sa promenom stanja u realnom sistemu

KLASIFIKACIJA MODELA PODATAKA

- Uzimajući u obzir karakteristike pojedinih modela podataka, ali i istorijski gledano, Modeli podataka se mogu klasifikovati u tri generacije:

- Modeli podataka I generacije
- Modeli podataka II generacije
- Modeli podataka III generacije

1. Modeli podataka I generacije

- Svaki konvencionalni programski jezik je zaseban model podataka.

2. Modeli podataka II generacije

- Funkcionalni model podataka
- Hijerarhijski model podataka
- Mrežni model podataka
- Relacioni model podataka

Zajedničko za sve modele podataka II generaci je da Postoje komercijalni SUBP zasnovani na modelima podataka II generacije

3. Modeli podataka III generacije

- Model Objekti-veze
- Semantički model podataka
- Prošireni relacioni model podataka

Za modele podataka III generacije - karakteristično (zajedničko): Ne postoje komercijalni SUBP zasnovani na modelima podataka III generacije.

Danas najrasprostranjniji model

- RELACIONI MODEL PODATAKA
- Neki komercijalni relacioni SUBP
  - ORACLE
  - MS SQL SERVER
  - DB II
  - POSTGRES
  - MySQL

RELACIONI MODEL PODATAKA

- Osnovni pojam relacionog modela je **relacija**.
- Relacija se može posmatrati sa dva aspekta: ***značenje*** i ***sadržaj***
- ***Značenje relacije*** naziva se ***intenzijom*** i formalno se iskazuje **šemom relacije.**
- Sadržaj relacije naziva se ***ekstenzijom***, a iskazuje se tabelom podataka ili **pojavom šeme relacije.**

Matematičko shvatanje pojma relacije

Potrebno je razmotriti dve definicije:

- ***Dekartov proizvod***
- ***Relacija r***

Dekartov proizvod

- Neka su D1, D2, ..., Dn konačani skupovi. Dekartov proizvod tih skupova u oznaci:

  D1 x D2 x . . . x Dn

- Definišemo kao skup uređenih n-torki:

  <d1,d2,…,dn>

Relacija r prema matematičkom shvatanju

- ***Relacija*** **r** prema matematičkom shvatanju na skupovima D1, D2, ..., Dn naziva se podskup Dekartovog proizvoda, odnosno

$$r \subseteq D1 \times D2 \times \ldots \times Dn$$

RELACIJA U RELACIONOM MODELU PODATAKA

- U relacionom modelu podataka relacija odgovara dvodimenzionalnoj tabeli u kojoj
  - svaki red sadrži jednu n-torku, a
  - svaka kolona elemente jednog domena

Relaciona šema (ŠEMA RELACIJE)

- **Šema relacije**, u oznaci:

  N(R, F)

je u opštem slučaju određena:

- Svojim nazivom – **N**
- Skupom naziva obeležR ={A1, A2, ..., An}, i
- Skupom ograničenja F ={O1, O2, ..., Om}

Skup ograničenja može biti zadat skupom ***funkcionalnih zavisnosti*** koje opisuju odnose između elemenata domena skupova X i Y koji su podskupovi skupa obeležja R ($X \subseteq R$ i $Y \subseteq R$).

RELACIONI MODEL PODATAKA
- Strukturna komponenta -

- **Primarni ključ**
- Svaka n-torka relacije je jedinstvena, što znači da postoji jedinstveni identifikator n-torki. Taj identifikator senaziva ***primarni ključ*** šeme relacije

Primarni ključ (1/3)

Da bi neki podskup obeležja X iz skupa obeležja R bio primarni ključ šeme relacije, mora zadovoljavati uslove:

- ***Jedinstvenosti –*** odnosno, ni u jednoj ekstenziji relacije ***r*** ne postoje dve n-torke koje imaju jednake vrednosti svih obeležja iz X.
- ***Minimalnosti –*** odnosno, ne postoji takav skup X', koji je podskup od X (X' $\subseteq$ X) koji bi ispunjavao uslov jedinstvenosti.

Primarni ključ (2/3)

- Uslov minimalnosti ima za cilj da broj obeležja u ključu svede na minimum, jer
- Svaki skup obeležja koji u sebi sadrži minimalni ključ jednoznačno određuje n-torku relacije.

Primarni ključ (3/3)

Ako se primarni ključ šeme relacije:

- Sastoji od samo jednog obeležja, kažemo da je **primarni ključ prost**.
- Sastoji od više obeležja kažemo da je **preimarni ključ složen**.

Ostali važni pojmovi strukturne komponente – RELACIONOG MODELA -

- Super ključ
- Kandidat za ključ
- Ključna obeležja
- Neključna obeležja
- Atomarne vrednosti

Super ključ

- Podskup obeležja u relacionoj šemi R, koji zadovoljava uslov jedinstvenosti, a ne zadovoljava uslov minimalnosti naziva se ***super ključ***

Kandidat ključa

- Jedna šema relacije može imati više obeležja, ili podskupova obeležja koji zadovoljavaju uslove
  - *jedinstvenosti* i
  - *minimalnosti.*
- Takva obeležja se nazivaju kandidatom ključa. U tom slučaju se jedno od tih obeležja bira za primarni ključ.

Ključna obeležja

- Obeležja koja ulaze u satav prima ključa nazivaju se ključna obeležja.
- Obeležja koja pripadaju primarnom ključu se u oznaci šeme relacije podvlače

Neključna obeležja

- Obeležja koja ne ulaze u niti u jedan od kandidata ključa nazivaju se ***neključna obeležja***

Atomane vrednosti

- U relacionom modelu podataka domeni obeležja su skupovi ***atomarnih vrednosti.***

Kakve su to atomarne vrednosti?

- Atomarnim vrednostima nazivamo one vrednosti koje nije moguće rastaviti (dekomponovati), a da time ne bude uništeno njihovo značenje. Na primer, ako se broj indeksa rastavi na skup znakova broj indeksa postaje izgubljen.

Šema relacione baze podataka

Kako je definisana **Šema baze podataka** u relacionom modelu podataka:

- Šema BP u RMP predstavlja konačan skup šema relacija.

$$S = \{Ni(Ri, Fi) \mid i = 1,k\}$$

- **Baza podataka –** Jedna pojava šeme baze podataka nad šemom S predstavlja skup pojava šema relacija koje čine šemu baze podataka.

PRIMER –Relacione- Šeme baze podataka

Na primer, neka imamo sledeću logičku šemu baze podataka:

NASTAVNIK (S_NAS, PREZIME_IME, ZVANJE,
S-DIR, DATZAP, PLATA, DODATAK)

PREDMET(S_PRED, NAZIV, MESTO)

PREDAJE (S_NAS, S_PRED, ČASOVA)

Ovo je **Logička šema baze podataka** – nezavisna od bilo kog SUBP.

Strukturna komponenta RELACIONOG MODELA

Spoljni ključ

- Može se primetiti da se ključevi šema relacija NASTAVNIK i PREDMET javljaju kao obeležja šeme relacije PREDAJE. Takva obeležja se nazivaju ***spoljni ključ.***

Šta je to Spoljni ključ** u **jednoj Šemi relacije?

- **Spoljni ključ -** je obeležje ili skup obeležja koji nisu primarni ključ u datoj već u nekoj drugoj šemi relacije. Takva obeležja mogu biti i deo ključa šeme ralacije kao šro je to slučaj sa obeležjima S_NAS i S_PRED u šemi relacije PREDAJE.

Strukturna komponenta RM - zavrsne napomene

- Šema relacije ne sme sadržavati dva jednaka obeležja (dva obeležja s istim nazivom).
- Različite šeme ralacije mogu sadržavati ista obeležja (obeležja sa istim nazivom) .
- Pri izvođenji operacija nad relacijama potrebno je da naziv svakog obeležja bude jednoznačno identifikovan.

Strukturna komponenta - zavrsne napomene -

- Obzirom da su nazivi šema relacija u šemi baze podataka jedinstveni, a da se u različitim šemama relacija mogu pojavljivati obeležja sa istim nazivima, ponekad je potrebno vršiti kvalifikaciju obeležja.

Šta je to kvalifikacija obeležja i kako se vrši?

- Ispred obeležja se navodi naziv šeme relacije kojoj obeležje pripada, na primer:

Predaje.S_Pred

- Kvalifikaciju obeležja treba vršiti samo kad u više šema relacija postoje isti nazivi obeležja.

OPERACIJSKA KOMPONENTA
- RELACIONOG MODELA -

- Operacijska komponenta omogućava manipulisanje podacima u bazi podataka, što se pre svega odnosi na:
  - postavljanje upita i
  - ažuriranje baze podataka.
- Smisao operacija nad relacionom bazom podataka sastoji se u promeni pojave šeme baze podataka ili u formiranju novih relacija.
- Pri tome se rezultat svakog upita posmatra kao formiranje nove relacije koja zadovoljava postavljene upite

OPERACIJSKA KOMPONENTA
- RELACIONOG MODELA -

- Svi postojeći jezici relacionih baza podataka, a najpoznatiji je SQL, razvijeni su na osnovu:
  - **relacione algebre**, ili
  - **relacionog računa**

Relaciona algebra

- je model proceduralnog jezika i sastoji se od skupa operatora za rad sa relacijama, odnosno od skupa operacija definisanih nad tim operatorima.
- Kombinovanjem operacija moguće je izvršiti pretraživanje, odnosno ažuriranje baze podataka.

Relacioni račun

- Zasnovan je na računu predikata prvog reda i spada u **neproceduralne jezike**.
- Pomoću relacionog računa definiše se traženi rezultat, a SUBP se prepušta da dođe do tog rezultata.
- Relacioni račun i relaciona algebra su međusobno ekvivalentni, odnosno bilo koji izraz relacionog računa može se transformisati u semantički ekvivalentan niz operacija relacione algebre.

RELACIONA ALGEBRA

- Operacije relacione algebre mogu se podeliti u dve grupe. Prvu grupu čine operacije matematičke teorije skupova:
  - UNIJA,
  - PRESEK,
  - RAZLIKA, i
  - DEKARTOV PROIZVOD,
- i u relacionom modelu podataka se mogu primeniti pošto je relacija definisana kao skup *n*-torki.

RELACIONA ALGEBRA

- Drugu grupu operacija čine operacije koje su posebno razvijene za relacioni model podataka:
  - SELEKCIJA,
  - PROJEKCIJA i
  - SPOJ.

RELACIONA ALGEBRA

- Za izdvajanje *n*-torki iz pojedinih relacija i njihovo eventualno kombinovanje sa *n*-torkama iz drugih relacija, da bi se definisao/specificirao upit, kao i za formalno iskazivanje ažuriranja relacione baze podataka koristi se relaciona algebra.
- Rezultat svake operacije nad nekom relacijom ili skupom relacija je nova relacija nad kojom se dalje mogu primenjivati operacije relacione algebre.

RELACIONA ALGEBRA

- Na primer, **ažuriranje relacione baze podataka** dodavanjem nove *n*-torke **može se formalno iskazati operacijom UNIJE** između relacije u bazi podataka i relacije koja sadrži samo *n*-torku koju treba dodati u bazu podataka.
- Kao posledica formalizma na kojem su zasnovani, jedna od glavnih osobina relacionih sistema je njihovo relativno jednostavno korišćenje.
- Interakcija korisnika sa sistemom odvija se na prirodan način korišćenjem neproceduralnog jezika kojim se specificiraju podaci koji se žele, a ne i kako se do tih podataka dolazi.

RELACIONA ALGEBRA

- Među jezicima relacionih baza podataka **SQL (*Structured Query Language*)** se nametnuo kao standard, tako da je danas sastavni deo najvećeg broja (gotovo svih) komercijalno raspoloživih sistema za upravljanje bazom podataka od personalnih do “velikih” računara.

Relaciona Algebra
- Binarne operacije -

  - ***unija,***
  - ***presek i***
  - ***razlika***
- Mogu se izvoditi samo na relacijama koje su međusobno uporedive.
- Relacije koje dozvoljavaju da se nad njima izvode operacije ***unije, preseka*** i ***razlike*** nazivaju se unijski kompatibilne relacije.
- **Dve relacije su unijski kompatibilne**,:
  - ako imaju isti broj obeležja,
  - ako imaju domene iste vrste.

Operacija UNIJE

- Neka su ***r*(R)** i ***p*(P)** dve unijski kompatibilne relacije.
- Unija relacija ***r*(R)** i ***p*(P)** u oznaci $r \cup p$ je skup n-torki ***t*** sadržanih u relaciji ***r,*** relaciji ***p*** ili u obe relacije, odnosno:

$$r \cup p = \{ t \mid t \in r \vee t \in p \}$$

- Prilikom izvođenja operacije unije potrebno je izvršiti usklađivanje redosleda kolona i naziva unijski kompatibilnih obeležja relacija ***r*** i ***p.***

Operacija UNIJE – PRIMER -

Primer: Operaciju unije na dve unijski kompatibilne relacije ***r*(R)** i ***p*(P)** koje imaju iste relacione šeme, odnosno R = P i usklađen redosled obeležja, ilustrovaćemo sledećim primerom :

r(A B C)	***p* (** A B C)	(***r*** ∪ ***p***) (A B C)
a b c	a b c	a b c
d e f	a b f	d e f
c b f		c b f
		a b f

Operacija RAZLIKA

- Razlika između dve unijski kompatibilne relacije ***r*(R)** i ***p*(P)** u oznaci ***r – p*** je skup n-torki sadržanih u relaciji ***r***, koje istovremeno nisu sadržane u relaciji ***p***, odnosno:

$$r - p = \{ t \mid t \in r \wedge t \notin p \}$$

- Operacija razlike nema osobinu komutativnosti i asocijativnosti.
- Kao i kod operacije unije i kod operacije razlike potrebno je pre izvođenja operacije izvršiti usklađivanje redosleda i naziva unijski kompatibilnih obeležja

Operacija RAZLIKA – PRIMER -

Primer: Neka su ***r*(R)** i ***p*(P)** dve unijski kompatibilne relacije i neka važi R = P.

r(A B C)	***p* (** A B C)	(***r* - *p***) (A B C)	(*p* **-** ***r***) (A B C)
a b c	a b c	d e f	a b f
d e f	a b f	c b f	
c b f			

Operacija PRESEK

- Operacija presek je binarna operacija koja kao i unija i razlika zahteva unijsku kompatibilnost relacija nad kojima se izvodi.
- Presek dve unijski kompatibilne relacije ***r*(R)** i ***p*(P)** je relacija koja sadrži sve n-torke koje su istovremeno elementi i relacije ***r*(R)** i relacije ***p*(P)**, odnosno:

$$r \cap p = \{ t \mid t \in r \wedge t \in p \}$$

- Operacija presek može biti izražena i pomoću operacije razlike.

$$r \cap p = r - (r - p)$$

Operacija PRESEK – PRIMER -

Primer: Neka su ***r*(R)** i ***p*(P)** dve unijski kompatibilne relacije i neka važi R = P.

r(A B C)
a b c
d e f
e f g
a b g

p **(** A B C)
a b c
a b f
e f g

(***r*** ∩ ***p***) = ***r'*** (A B C)
a b c
e f g

Operacija DEKARTOV PROIZVOD

- Dekartov proizvod dve relacije ***r*(R)** i ***p*(P)** u oznaci ***r* x *p*** predstavlja skup n-torki koje su nastale kao rezultat spajanja (konkatenacije) svake pojedine *n*-torke sadržane u relaciji ***r*** sa svakom pojedinom *n*-torkom iz relacije ***p***, odnosno:

$$r \times p = \{ (tr, tp) \mid tr \in r \wedge tp \in p \}$$

- Relaciona šema na kojoj je zadata relacija dekartovog proizvoda ima za obeležja uniju skupova obeležja relacionih šema **R** i **P.**

Operacija DEKARTOV PROIZVOD

Primer: Neka su ***r*(R)** i ***p*(P)** relacije na relacionim šemama **R** i **P.**

R(A B)
a b
c d
a c

p **(**C D)
b c
d e

(***r x p***) (A B C D)
a b b c
a b d e
c d b c
c d d e
a c b c
a c d e

Operacija SELEKCIJA

- Selekcija je unarna operacija kojom se iz relacije izdvaja određeni podskup *n*-torki. U naredbi mora biti definisan uslov F, ili kriterijum na osnovu kojeg se vrši izdvajanje *n*-torki.
- Selekcija na relaciji ***r***, prema uslovu, F je relacija koja sadrži sve *n*-torke sadržane u ***r*** koje zadovoljavaju uslov F, odnosno:

$$\sigma F\,(r) = \{\, t \mid t \in r \wedge \ t \text{ zadovoljava } F \,\}$$

Operacija SELEKCIJA

Kriterijum za selekciju F može sadržavati:

- konstante,
- nazive obeležja relacije nad kojom se selekcija izvodi,
- aritmetičke operatore poređenja ($=$, $\neq$, $<$, $\leq$, $>$, $\geq$)
- aritmetičke operatore (+, -, *, /)
- logičke operatore ($\vee$, $\wedge$, $\neg$).
- Zagrade ( ) za izmenu redosleda operacija.

Operacija SELEKCIJA

Primer: Nekaje relacija ***r*(R)** sledećeg oblika:

r(A B C)	**σA=a (*r*) (** A B C)	**σB=C (*r*)** (A B C)	**σB<C (*r*)** (A B C)
a 1 2	a 1 2	a 5 5	a 1 2
d 3 4	a 5 5		d 3 4
a 5 5			

Operacija PROJEKCIJA

- Projekcija je unarna operacija kojom se iz date relacije izdvajaju pojedine kolone.
- Neka su X i R skupovi obeležja i neka važi $X \subseteq R$. Neka je ***r*(R)** relacija zadata na skupu obeležja R. Projekciju relacije ***r*** na skup obeležja X predstavlja relaciju koja se označava sa πX (*r*) i definiše kao:

$$\pi X\ (r) = \{\ t[X] \mid t \in r\ \}$$

Operacija PROJEKCIJA

Treba obratiti pažnju da u izrazu **t[X]:**

[X] – predstavlja operator koji izdvaja skup vrednosti (znači [ ] nisu obične zagrade nego predstavljaju operator).

t[X] – predstavlja skup vrednosti koje skup obeležja X uzima u relaciji r

Operacija PROJEKCIJA – PRIMER -

Primer: Neka je ***r*(R)** relacija na relacionoj šemi R koja sadrži obeležja A, B, i C. Projekcija relacije *r* na obeležjima A i B je

r(A B C)	**(** A B)	**πAB (*r*) = *r'*** (A B)
a b c	a b	a b
d e f	d e	d e
a b f	a b	

Operacija SPOJ

Operacija spoja (*join*) je složena binarna operacija za koju se može reći da se izvodi u tri koraka:

1. Korak – formiranje dekartovog proizvoda relacija;
2. Korak – iz dekartovog proizvoda se izdvajaju *n*-torke koje zadovoljavaju postavljene uslove
3. Korak – iz tabele dobijene u drugom koraku izdvajaju se određene kolone. Ovaj korak obavezno seizvodi se samo u slučaju prirodnog spoja.

Operacija SPOJ – PRIMER -

- **Primer:** Neka su ***r*(R)** i ***p*(P)** relacije i neka obeležje relacije B pripada šemi relacije R, a obeležje C šemi relacije P. Operacija spoja relacija *r* i *p* na osnovu uslova B < C ima sledeći tok.

r(A B)	***p*** **(** C D)
a 1	2 d
e 3	1 f
f 1	

Operacija SPOJ – PRIMER -

1. Korak: (***r*** **x** ***p***) = ***r'*** (A B C D)

a	1	2	d
a	1	1	f
e	3	2	d
e	3	1	f
f	1	2	d
f	1	1	f

2. Korak: **σB < C (*r'*) = *r''* (** A B C D)

a	1	2	d
f	1	2	d

Operacija SPOJ

- *Theta spoj* (θ - spoj) predstavlja najopštiji oblik operacije spoja. Neka su ***r*(R)** i ***p*(P)** relacije i neka važi Ai $\in$ R i Bj $\in$ P. Theta spoj relacija *r* i *p* na osnovu uslova izdvajanja Ai θ Bj (pri čemu $\theta \in \{=, \neq, <, \leq, >, \geq\}$) u oznaci *r*[Ai θ Bj] je skup *n*-torki koji zadovoljava sledeće uslove:
- podskup je dekartovog proizvoda relacija *r* i *p*, i
- svaki element tog podskupa zadovoljava uslov izdvajanja [Ai θ Bj], odnosno:

$$r\,[Ai\ \theta\ Bj]\,p = \{\ (\ tr, tp\ )\ |\ tr \in r \wedge tp \in p\ \wedge tr\,[Ai]\ \theta\ tp[Bj]\ \}$$

Operacija SPOJ

- Operacija θ - spoja nad relacijama *r*(R) i *p*(P) je relacija ako je i Dekartov proizvod nad istim relacijama relacija.
- Treba se podsetiti uslova $R \cap P = \varnothing$
- U tom slučaju operaciju θ -spoja možemo formalno definisati na osnovu operacija Dekartovog proizvoda i selekcije:
- $\boldsymbol{r\,[Ai\ \theta\ Bj]\,p = \sigma_{(Ai\ \theta\ Bj)}\,(rxp)}$

Operacija SPOJ

- Postoje dva posebna slučaja θ spoja:
  - kad uslov izdvajanja ne postoji; tada operacija θ - spoja prelazi u Dekartov proizvod.
  - slučaj kod kojeg je θ operator znak jednakosti (=), koji se naziva spoj sa izjednačavanjem (*equi-join)*
- Rezultat operacije spoja sa izjednačavanjem je tabela sa dve potpuno identične kolone.
- Obeležja čije vrednosti su sadržane u tim kolonama mogu ali ne moraju biti različita

Operacija SPOJ

- **Primer:** Neka su ***r*(R)** i ***p*(P)** relacije, pri čemu važi B $\in$ R i C $\in$ P, a uslov izdvajanja je B = C.

r (A B)

a	3
b	2
e	4

p (C D)

2	d
3	a

r [B = C] *p* = *q* (A B C D)

a	3	3	a
b	2	2	d

Prirodni spoj

- ***Prirodni spoj*** – je u tesnoj vezi sa spojem sa izjednačavanjem.
- Kao rezultat θ spoja sa izjednačavanjem dobija se tabela s najmanje dve kolone identičnih vrednosti obeležja, a često su i nazivi obeležja identični
- Da bi se tabela koja sadrži po dve kopije jednog ili više obeležja transformisali u relaciju, moraju se odstraniti obeležja iz zaglavlja tabele zajedno sa odgovarajućim kolonama u tabeli.
- Ova dodatna operacija izvodi se u okviru operacije prirodnog spoja.

Operacija Prirodnog spoja

- **Prirodnim spojem dve relacije spajaju se međusobno *n*-torke tih relacija na osnovu vrednosti obeležja koja se nalaze u obe šeme relacija.**

Neka su *r*(R) i *p*(P) relacije i neka je R ∪ P = T. Prirodno spajanje relacija *r* i *p* u oznaci *r* ►◄ *p*, kao rezultat daje relaciju *q*(T) .

- **Za svaku *n*-torku *tq* u relaciji *q* postoje *n*-torke *tr* ∈ *r* i *tp* ∈ *p* za koje važi tr = tq[R] i tp = tq[P] odnosno:**

$$r \;►◄\; p = \{\, tq \mid tq[R] = tr \wedge tr \in r \wedge tq[P] = tp \wedge tp \in p \,\}$$

Operacija Prirodnog spoja

Operaciju prirodnog spoja možemo izraziti pomoću jednostavnih operacija:

- Dekartovog proizvoda,
- selekcije, i
- projekcije
- Neka su *r*(R) i *p*(P) relacije, neka je $R \cap P = X$ i neka važi $R.X \in R$ i $P.X \in P$. Tada važi:

$$r \blacktriangleright\blacktriangleleft p = \pi_{RP}\, \sigma_{R.X = P.X}\, (r \times p)$$

Operacija Prirodnog spoja

- **Primer:** Neka su ***r*(R)** i ***p*(P)** relacije i neka važi R= { A, B, C}, P = {B, C, D} i T = { A, B, C, D}

r (A B C)			*p* (B C D)			*r* ►◄ *p* = *t* (A B C D)			
d	1	6	1	6	k	d	1	6	k
b	3	6	3	6	h	b	3	6	h
c	2	7	4	8	k				

INTEGRITETNA KOMPONENTA

- Model baze podataka polazi od pretpostavke da su sve relevantne informacije poznate i da se mogu uneti u bazu podataka.

- U praksi ova pretpostavka često nije ispunjena. Ovaj problem se naziva ***problem nedostajućih informacija***.

Nedostajuce informacije u BP

- Problem nedostajućih informacija se pojavljuje, na primer kada u trenutku unosa podataka u bazu podataka nisu poznati ***datum rođenja, adresa stanovanja osobe*** i slično.
  - Uz određeni trud te informacije bi se eventualno mogle pribaviti.
  - Postoje međutim slučajevi kada bez obzira na potreban trud nije moguće doći do određenih informacija. Na primer, nije moguće utvrditi ime bračnog druga osobe koja nije u braku.

NULL - Vrednosti

- Na mesto na koje treba upisati stvarnu vrednost obeležja koje nam nije poznato, u bazu podataka se upisuje ***null – vrednost***
- ***Null – vrednost*** predstavlja oznaku da stvarna vrednost nije poznata.
- Uvođenje null – vrednosti komplikuje operacijsku komponentu relacionog modela, pa ih treba izbegavati

INTEGRITETNA KOMPONENTA – RMP -

Šta je u najširem smislu Integritet baze podataka?

- Pod integritetom baze podataka podrazumeva se ispravnost i istinitost podataka sadržanih u bazi podataka.
- Uslovima ili pravilima integriteta se definišu ograničenja sadržaja baze podatak na neka dozvoljena stanja.

INTEGRITETNA KOMPONENTA - RMP

Uslove integriteta koji se javljaju u jednoj relacionoj bazi podataka možemo podeliti u dve grupe:

- **Opšti uslovi integriteta** – važe u svim relacionim bazama podataka
- **Korisnička pravila integriteta** – specifična za pojedine aplikativne sisteme, odnosno oblasti primene.

OPŠTI USLOVI INTEGRITETA RBP

Relacioni model podataka poseduje dva opšta uslova integriteta:

- **integritet entiteta, i**
- **referencijalni integritet**

Integritet entiteta

Integritet entiteta – vezan je za pojam primarnog ključa šeme relacije. Obzirom na osobine primarnog ključa sledi i definicija integriteta entiteta, koja glasi:

- **Vrednost primarnog ključa kao celine i niti jedne njegove komponente ne sme biti jednaka null – vrednosti.**

Primarni ključ omogućava jednostavno i efikasno adresiranje *n*-torki. *Null*-vrednost primarnog ključa kao celine ili neke njegove komponente ne bi mogla ostvariti prethodnu ulogu.

Referencijalni integritet

Referencijalni integritet predstavlja poseban slučaj opštijeg uslova integriteta, koji se naziva zavisnost sadržavanja.

- Zavisnost sadržavanja se naziva izraz oblika:

$$R[Y] \subseteq P[X]$$

- gde su R i P dve šeme relacija, a $Y \subseteq R$ i $X \subseteq P$ skupovi unijski kompatibilnih obeležja.
- Zavisnost sadržavanja $R[Y] \subseteq P[X]$ je zadovoljena, ako važi:

$$(\forall\ \mathbf{tr} \in \boldsymbol{r}(\mathbf{R})\ )\ (\ \exists\ \mathbf{tp} \in \boldsymbol{p}(\mathbf{P})\ |\ \mathbf{tr[Y] = tp[X]}\ )$$

Referencijalni integritet

- Zavisnost sadržavanja definiše egzistencijalno ograničenje u smislu da se u relaciju *r* ne može upisati *n*-torka tr (***pozivajuća n-torka***), ako u relaciji *p* ne postoji bar jedna *n*-torka tp (***ciljna n-torka***) takva da važi tr[Y] = tp[X].
- Pri brisanju *n*-torke tp iz *p*, ako važi ($\exists$ tp $\in$ *p*(P) | tr[X] = x), moraju se brisati sve *n*-torke tr iz *r*(R) za koje važi tr[Y] = x.

Referencijalni integritet

NASTAVNIK (S-NAS, PREZIME-IME, ZVANJE, S-KAT,....)
(*pozivajuća*)

KATEDRA (S-KAT, NAZIV-KAT, ...) (*ciljna*)

Standardni upitni jezk – SQL

SQL nije samo upitni jezik, već predstavlja kompletan jezik podataka koji sadrži jezike i za:

- definisanje podataka,
- ažuriranje,
- kontrolu konzistentnosti,
- konkurentni rad, i
- jezik za održavanje rečnika podataka.

Ugrađen je u većinu komercijalno raspoloživih SUBP, od personalnih do velikih računara.

Standardni upitni jezk – SQL

Osnovne karakteristike SQL-a su:

- **Jednostavnost i jednoobraznost pri korišćenju** (relacije se kreiraju jednom izvršnom naredbom);
- **Mogućnost interaktivnog i klasičnog programiranja** (koristeći SQL dobijaju se odgovori na na trenutno postavljene zahteve ili se SQL blokovi ugrađuju u neki viši programski jezik);
- **Neproceduralnost** (odnosno, proceduralnost u minimalnom stepenu; SQL – je u velikoj meri neproceduralan jer se njime definiše **ŠTA** se želi dobiti, a **ne kako** se do rezultata dolazi).

SQL naredbe

SQL naredbe mogu se grupisati u više grupa. Mi ćemo obraditi dve grupe naredbi SQL-a:

- **Naredbe za definisanje podataka:**
- **Naredbe za manipulisanje podacima:**

SQL Naredbe za definisanje podataka:

- Kreiranje relacije (CREATE TABLE)
- Kreiranje pogleda (CREATE VEW)
- Promena strukture relacije (ALTER TABLE)
- Brisanje relacije iz baze podataka (DROP TABLE)
- Kreiranje indeksa (CREATE INDEX)
- Brisanje indeksa iz baze podataka (DRO INDEX)

SQL Naredbe za manipulisanje podacima:

- Dodavanje novih *n*-torki u relaciju (INSERT)
- Pretraživanje relacione baze podataka (SELECT)
- Izmena sadržaja relacije (UPDATE)
- Brisanje *n*-torki relacije (DELETE)

SQL – CREATE naredba

Naredba za definisanje tabele (šeme relacije)

Osnovni oblik sintakse

CREATE TABLE <naziv_tabele>

(naziv_obeležja> <tip_podatka> [<granicenje_na_obeležje>]

[, naziv_obeležja> <tip_podatka> [<granicenje_na_obeležje>]]

. . . .

[<ograničenja_na_tabelu>]

)

SQL – CREATE naredba

CREATE TABLE
[*database_name*.[*owner*] . | *owner*.] *table_name*
({ < column_definition >
| *column_name* AS *computed_column_expression*
| < table_constraint > ::= [CONSTRAINT *constraint_name*] }
| [{ PRIMARY KEY | UNIQUE } [,...*n*]
)

[ON { *filegroup* | DEFAULT }]
[TEXTIMAGE_ON { *filegroup* | DEFAULT }]

Definicija kolone (obeležja)

< column_definition > ::= { *column_name data_type*
}
[COLLATE < collation_name >]
[[DEFAULT *constant_expression*]
| [IDENTITY [(*seed* , *increment*) [NOT
FOR REPLICATION]]]
]
[ROWGUIDCOL]
[< column_constraint >] [...*n*]

Ograničenje na kolonu (obeležje)

- < column_constraint > ::= [CONSTRAINT *constraint_name*]
  { [NULL | NOT NULL]
  | [{ PRIMARY KEY | UNIQUE }
  [CLUSTERED | NONCLUSTERED]
  [WITH FILLFACTOR **=** *fillfactor*]
  [ON {*filegroup* | DEFAULT}]]
  ]
  | [[FOREIGN KEY]
  REFERENCES *ref_table* [**(** *ref_column* **)**]
  [ON DELETE { CASCADE | NO ACTION }]
  [ON UPDATE { CASCADE | NO ACTION }]
  [NOT FOR REPLICATION]
  ]
  | CHECK [NOT FOR REPLICATION]
  (*logical_expression* **)**
  }

Ograničenje na tabelu

< table_constraint > ::=
[CONSTRAINT *constraint_name*]
{ [{ PRIMARY KEY | UNIQUE }
[CLUSTERED | NONCLUSTERED]
{ **(** *column* [**,**...*n*] **)** }
[WITH FILLFACTOR **=** *fillfactor*]
[ON {*filegroup* | DEFAULT }]
]
| FOREIGN KEY
[**(** *column* [**,**...*n*] **)**]
REFERENCES *ref_table* [**(** *ref_column* [**,**...*n*] **)**]
[ON DELETE { CASCADE | NO ACTION }]
[ON UPDATE { CASCADE | NO ACTION }]
[NOT FOR REPLICATION]
| DEFAULT *constant_expression*
[FOR *column*] [WITH VALUES]
| CHECK [NOT FOR REPLICATION]
(*search_conditions* **)**

SELECT – SQL Naredba

Retrieves rows from the database and allows the selection of one or many rows or columns from one or many tables. The full syntax of the SELECT statement is complex, but the main clauses can be summarized as:

SELECT *select_list*
[INTO *new_table*]
FROM *table_source*
[WHERE *search_condition*]
[GROUP BY *group_by_expression*]
[HAVING *search_condition*]
[ORDER BY *order_expression* [ASC | DESC]]

Ograničenje na tabelu (relaciju)

< table_constraint > ::= [CONSTRAINT *constraint_name*]
{ [{ PRIMARY KEY | UNIQUE }
[CLUSTERED | NONCLUSTERED]
{ **(** *column* [ASC | DESC] [**,**...*n*] **)** }
[WITH FILLFACTOR **=** *fillfactor*]
[ON { *filegroup* | DEFAULT }]
]
| FOREIGN KEY
[**(** *column* [**,**...*n*] **)**]
REFERENCES *ref_table* [**(** *ref_column* [**,**...*n*] **)**]
[ON DELETE { CASCADE | NO ACTION }]
[ON UPDATE { CASCADE | NO ACTION }]
[NOT FOR REPLICATION]
| CHECK [NOT FOR REPLICATION]
(*search_conditions* **)**
}

CREATE INDEX – SQL Naredba

CREATE [UNIQUE] [CLUSTERED |
NONCLUSTERED] INDEX *index_name*
ON { *table* | *view* } **(** *column* [ASC | DESC
] [**,**...*n*] **)**
[WITH < index_option > [**,**...*n*]]
[ON *filegroup*]

Opcije INDEXA

- < index_option > :: =
  { PAD_INDEX |
  FILLFACTOR **=** *fillfactor* |
  IGNORE_DUP_KEY |
  DROP_EXISTING |
  STATISTICS_NORECOMPUTE |
  SORT_IN_TEMPDB
  }

Opšti oblik naredbe za kreiranje INDEKSA

CREATE [UNIQUE] INDEX <Naziv_indeksa>

ON <Naziv_Relacije (Obelezje1 [ASC | DESC] [, Obelezje2 [ASC | DESC]] . . .]

[<Ostali_Parametri>]

Primer: Indeksi

Kreirati indeks nad relacijom Nastavnik po obeležju Prezimre_Ime.

```
Create Index Abc_Ime
    On Nastavnik(Prezime_Ime)
```

DROP INDEX – SQL Naredba

DROP INDEX '*table*.*index* | *view.index*' [,...*n*]

Dodavanje Novih n-torki u relaciju - INSERT

Sintaksa:

INSERT INTO <Naziv_Relacije>
[(Obl1, Obl2, …)]
{VALUES (<vrednost1>, <vrednost2>,…) |
<podupit>}

Primer – INSERT Naredba

Treba dodati podatke u relaciuju Nastavnik za Petrovic Petra sa Sifrom 002, zvenjem DOCENT koji radi od 1. februara 1982, prima platu 11500, nema dodataki kojem je rukovorilac Nastavnik sa sifrom 001.

```
Insert Into Nastavnik Values
   (002,'PETROVIC PETAR', 'DOCENT',006,'1982-02-01',
       11500,Null)
```

SELECT – SQL Naredba

- Retrieves rows from the database and allows the selection of one or many rows or columns from one or many tables. The full syntax of the SELECT statement is complex, but the main clauses can be summarized as:

SELECT *select_list*
[INTO *new_table*]
FROM *table_source*
[WHERE *search_condition*]
[GROUP BY *group_by_expression*]
[HAVING *search_condition*]
[ORDER BY *order_expression* [ASC | DESC]]

SELECT Klauzula

Specifies the columns to be returned by the query.

Syntax

```
SELECT [ ALL | DISTINCT ]
        [ TOP n [ PERCENT ] [ WITH TIES ] ]
        < select_list >


< select_list > ::=
   {    *
            | { table_name | view_name | table_alias }.*
            |     { column_name | expression | IDENTITYCOL |
    ROWGUIDCOL }
                  [ [ AS ] column_alias ]
            | column_alias = expression
        }    [ ,...n ]
```

FROM Klauzula

Specifies the table(s) from which to retrieve rows. The FROM clause is required except when the select list contains only constants, variables, and arithmetic expressions (no column names).

Syntax

[FROM { < table_source > } [,...*n*]]

< table_source > ::=
**   *table_name* [[AS] *table_alias*] [WITH (< table_hint > [,...*n*])]**
** | *view_name* [[AS] *table_alias*]**
** | *rowset_function* [[AS] *table_alias*]**
** | OPENXML**
** | *derived_table* [AS] *table_alias* [(*column_alias* [,...*n*])]**
** | < joined_table >**

< joined_table > ::=
**   < table_source > < join_type > < table_source > ON < search_condition >**
** | < table_source > CROSS JOIN < table_source >**
** | < joined_table >**

< join_type > ::=
**   [INNER | { { LEFT | RIGHT | FULL } [OUTER] }]**
**   [< join_hint >]**
**   JOIN**

Obavezni elementi SELECT Naredbe

SELECT { * | {<NazivRelacije>.* | element_selekcije}}
    FROM <NazivRelacije>

Prikaz kompletnog sadržaja relacije

Primer: Prikazati kompletan sadržaj relacija: Nastavnik, Predmet i Predaje

```
Select * From Nastavnik
```

Selekcija željenih kolona

Primer: Prikazati S_Nas, Prezime_Ime iz relacije Nastavnik

```
Select S_nas As Sifra, Prezime_Ime
  From Nastavnik
```

Selekcija željenih n-torki Klauzula WHERE

Primer: Prikazati sve predmete koji se predaju u 8. semestru.

WHERE Clause

Specifies a search condition to restrict the rows returned.

Syntax

[WHERE < search_condition > | < old_outer_join >]

< old_outer_join > ::=

column_name { * = | = * } *column_name*

ALTER TABLE – SQL Naredba

```
ALTER TABLE table
    { [ ALTER COLUMN column_name
        { new_data_type [ ( precision [ , scale ] ) ]
            [ COLLATE < collation_name > ]
            [ NULL | NOT NULL ]
            | {ADD | DROP } ROWGUIDCOL }
        ]
        | ADD
            { [ < column_definition > ]
            |  column_name AS computed_column_expression
            } [ ,...n ]
        | [ WITH CHECK | WITH NOCHECK ] ADD
            { < table_constraint > } [ ,...n ]
        | DROP
            { [ CONSTRAINT ] constraint_name
                | COLUMN column } [ ,...n ]
        | { CHECK | NOCHECK } CONSTRAINT
            { ALL | constraint_name [ ,...n ] }
        | { ENABLE | DISABLE } TRIGGER
            { ALL | trigger_name [ ,...n ] }
    }
```

DROP TABLE – SQL Naredba

Removes a table definition and all data, indexes, triggers, constraints, and permission specifications for that table. Any view or stored procedure that references the dropped table must be explicitly dropped by using the DROP VIEW or DROP PROCEDURE statement.

Syntax

DROP TABLE *table_name*