SQL

Structured Query Language

Prof dr. Borivoje Milošević


Spojene Relacije
Definicija Podataka
Modifikacije Baze
Osnovna Struktura Upita
SQL
Skupovne Operacije
Kompleksni Upiti
Agregatne Funkcije
Null Vrednosti
Pogledi

SQL

- Konstrukcije relacione algebre i relacionog računa su za korisnika dosta složene. Zato SUBP sadrži upitne jezike koji su korisniku bliži i prihvatljiviji. Ti upitni jezici bliski su čovekovom prirodnom jeziku (engleskom), za razliku od izraza relacione algebre i relacionog računa.

- Najpoznatiji relacioni upitni jezici su:

  - **SQL** se temelji na kombinacijama relacione algebre i relacionog računa
  - **QUEL** primena računa n-torki
  - **QBE** – primena računa domena.

SQL

- SQL je počeo razvoj u IBM-u 70-ih godina 20.st. pod imenom SEQUEL godine (kratica engl. riječi Structured English Query Language). Danas se SQL jezik rabi u većini relacijskih baza podataka

- SQL omogućava korisnicima pristup podacima u sistemima za upravljanje relacionim bazama podataka, kao što su Oracle, Sybase, Informix, Microsoft SQL Server, Access i drugi, tako što dopušta korisnicima da opišu podatke koje žele da dobiju. SQL također omogućava korisnicima da definišu podatke u nekoj bazi podataka i da manipulišu tim podacima.

Istorijat

- IBM Esquel jezik razvijen kao deo projekta System R u IBM San Jose Research Laboratory
- Preimenovan u Structured Query Language (SQL)
- ANSI i ISO standard SQL:
- – SQL-86
- – SQL-89
- – SQL-92
- – SQL:1999 (usaglašen sa Y2K)
- – SQL:2003
- Komercijalni sistemi nude najveći deo ili sve mogućnosti standarda SQL-92, plus različite mogućnosti iz kasnijih standarda plus specijalne mogućnosti.

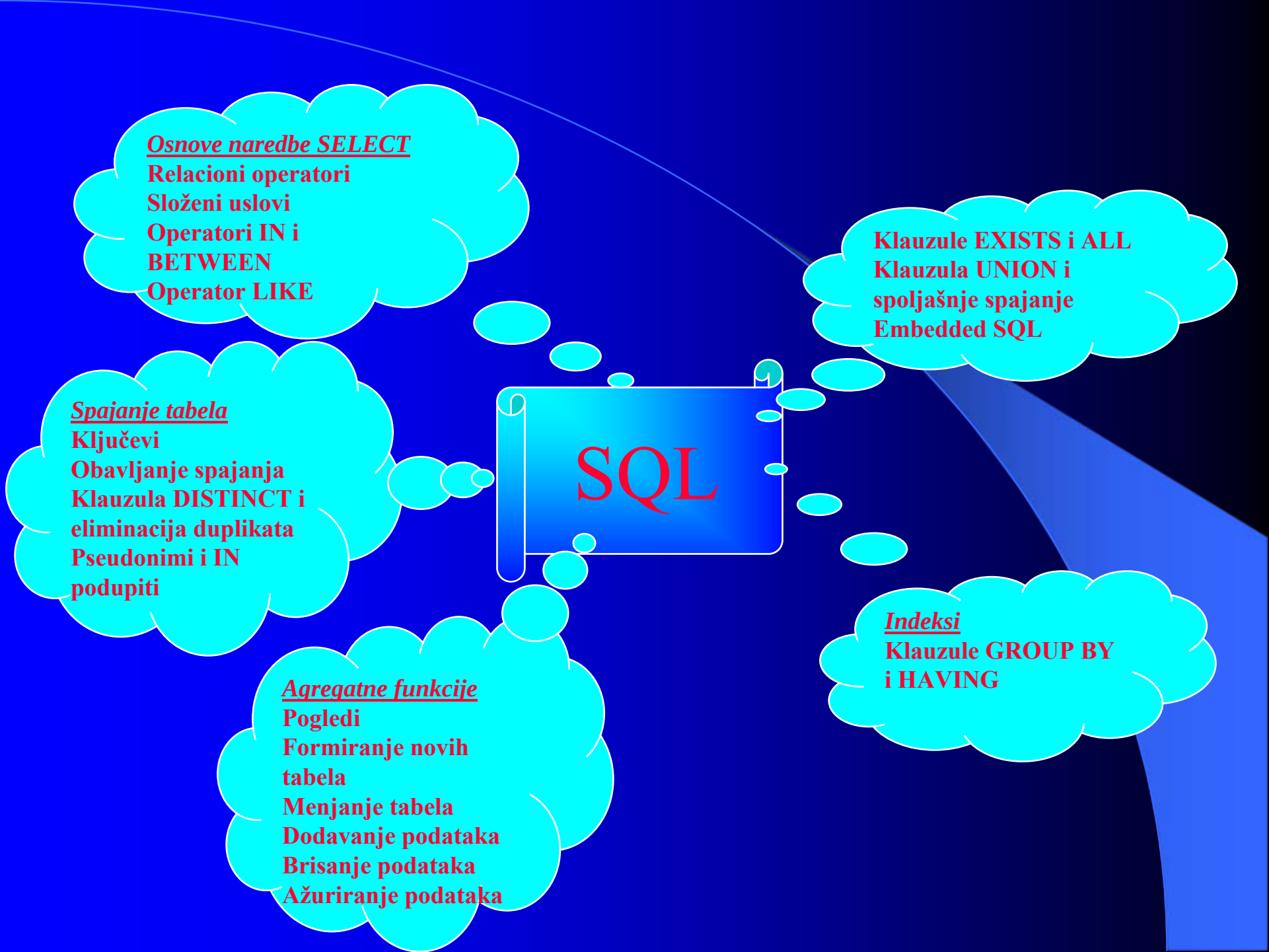
SQL

- SQL je nazvan jezikom iako on nije potpun programski jezik.

  Npr. u njemu se ne nalazi If...Then...Else konstrukcija za ispitivanje uslova, a nema ni konstrukciju za logičku strukturu petlje Do...While ili For...Next.

  On sadrži:

  - konstrukcije slične relacionoj algebri ili računu (koje su osnove jezika za rukovanje podacima),
  - naredbe opisa baze podataka (jezik za opis podataka),
  - naredbe za povezivanje SQL-a sa nekim standardnim programskim jezikom (CURSOR operacije), kao i
  - naredbe za definisanje zaštite i integriteta baze podataka, upravljanje konkurentnim obradama i oporavak baze podataka.


Osnove naredbe SELECT
Relacioni operatori
Složeni uslovi
Operatori IN i BETWEEN
Operator LIKE
Klauzule EXISTS i ALL
Klauzula UNION i spoljašnje spajanje
Embedded SQL
Spajanje tabela
Ključevi
Obavljanje spajanja
Klauzula DISTINCT i eliminacija duplikata
Pseudonimi i IN podupiti
SQL
Indeksi
Klauzule GROUP BY i HAVING
Agregatne funkcije
Pogledi
Formiranje novih tabela
Menjanje tabela
Dodavanje podataka
Brisanje podataka
Ažuriranje podataka

SQL

Zašto SQL?

- Najčešći i najuticajniji komercijalni jezik upita
- Standard u relacionim bazama podataka
- Razvijen u IBM Research Lab u San Jose u ranim 70-im

Osnovne naredbe:

- **select**
- **insert**
- **delete**
- **update**

Napomena: MS Access SQL pojednostavljena je verzija "standardnog" SQL-a

SQL

Osigurava naredbe za različite zadatke uključujući:

- **Upite nad podacima**
- **Dodavanje, menjanje i brisanje redova u tablicama**
- **Kreiranje, menjanje i brisanje objekata šeme**
- **Kontrolu pristupa bazi podataka i objektima šeme**
- **Osigurava konzistentnost baze podataka**

SQL izrazi - razvrstavanje

- **DML izrazi (Data Manipulation Language Statements) tj. izrazi za upravljanje podacima**
- **DDL izrazi (Data Definition Language Statements) tj. izrazi za definisanje podataka**
- **Izrazi za kontrolu transakcija (Transaction Control Statements)**
- **Izrazi za kontrolu sesije (Session Control Statements)**
- **Izrazi za kontrolu sistema (System Control Statements)**
- **Ugrađeni SQL izrazi (Embedded SQL Statements)**

DML izrazi

Koriste se za izvođenje sledećih akcija nad podacima:

- **Upisivanje redova u tabelu ili indeks (INSERT)**
- **Promena vrednosti u kolonama tabela ili pregleda (UPDATE)**
- **Pretraživanje redova iz tabela ili pregleda (SELECT)**
- **Brisanje redova iz tabela ili indeksa (DELETE)**
- **Omogućavanje pregleda izvođenja SQL izraza (EXPLAIN PLAN)**
- **Zaključavanje tabela i pregleda (LOCK TABLE)**

DDL izrazi

Kreiraju, menjaju i brišu objekte šeme:

- **Kreiranje, menjanje i brisanje objekata šeme i drugih struktura baze podataka (same baze podataka, korisnika nad bazom, tablica…)**
  (CREATE, ALTER i DROP)
- **Menjanje imena objekata šeme (RENAME)**
- **Praćenje statistike o objektima šeme, validiranje strukture objekata (ANALYSE)**
- **Davanje i uzimanje privilegija i uloga nad bazom (GRANT i REVOKE)**

Izrazi za kontrolu transakcije

Upravljaju promenama uzrokovanim DML izrazima:

- **Trajno potvrđivanje transakcije (COMMIT)**
- **Vraćanje baze u stanje pre izvođenja transakcije (ROLLBACK)**
- **Definisanje tačke od koje se može povratiti stanje baze (SAVEPOINT)**
- **Definisanje svojstava transakcije (SET TRANSACTION)**

CREATE TABLE

- Definisanje nove relacije, odnosno opis njene relacione šeme (tabele). U proširenoj sintaksi moguće je definisati ograničenja (CONSTRAINT)
- PRIMARY KEY(primarni ključ tabele)
- UNIQUE (jedinstveni ključ tabele)
- FOREIGN KEY (strani ključ tabele, referencijalni integritet) definiše se atribut (ili skup atributa) posmatrane tabele koji se referenciraju na primarni ključ iste ili neke druge tabele

```
CREATE TABLE grad(
pbr SMALLINT,
naziv VARCHAR(50),
CONSTRAINT grad_pk PRIMARY KEY(pbr) );

CREATE TABLE stanovnici(
jmbg INT,
ime osobe VARCHAR(30) NOT NULL,
prezime osobe VARCHAR(30) NOT NULL,
pbr SMALLINT,
adresa VARCHAR(100) NOT NULL,
CONSTRAINT stanovnici_pk PRIMARY
KEY(jmbg),
CONSTRAINT stanovnici_fk_grad FOREIGN
KEY(pbr)
REFERENCES grad(pbr));
```

FOREIGN KEY u jednoj tabeli ukazuje na PRIMARY KEY u drugoj tabeli:

"Persons" tabela

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

" Orders" tabela:

O_Id	OrderNo	P_Id
1	77895	3
2	44678	3
3	22456	2
4	24562	1

Primećujemo da "P_Id" kolona u tabeli "Orders" ukazuje na "P_Id" kolonu u tabeli "Persons" .

"P_Id" kolona u tabeli "Persons" je njen PRIMARY KEY.

"P_Id" kolona u tabeli "Orders" je njen FOREIGN KEY.

DROP TABLE

Naredba za uklanjanje (brisanje) relacije - tabele iz baze podataka

- Za razliku od DELETE koja izbacuje samo n-torke iz relacije, ova naredba izbacuje i definiciju relacije pa relacija i njena relaciona šema više ne postoji

- Sintaksa:

DROP TABLE *table_name*

- Primer:

DROP TABLE osobe

ALTER TABLE

Nareba za izmjenu definicije postojeće relacije

Dodavanje atributa:

- ALTER TABLE stanovnici ADD dat_rod DATETIME;

Uklanjanje atributa:

- ALTER TABLE radno_mesto DROP COLUMN broj_zaposlenih;

Izmena postojećih atributa:

- ALTER TABLE racuni ALTER COLUMN nacin_placanja CHAR(1);

ALTER TABLE

SQL ALTER TABLE primer

Posmatraćemo"Persons" tabelu:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

Želimo da dodamo kolonu koja se zove "DateOfBirth" u tabelu "Persons" .
KORISTIĆEMO SLEDEĆU SQL NAREDBU:

ALTER TABLE Persons ADD DateOfBirth date

P_Id	LastName	FirstName	Address	City	DateOfBirth
1	Hansen	Ola	Timoteivn 10	Sandnes	
2	Svendson	Tove	Borgvn 23	Sandnes	
3	Pettersen	Kari	Storgt 20	Stavanger	

GRANT

Vlasnik relacije je uvek korisnik koji je nju definisao naredbom CREATE TABLE, a pravo na izvršavaje SQL naredbi i kreiranje objekata vlasnik na druge osobe prenosi naredbom GRANT

Za naredbe DDL-a

Primeri:

- **GRANT CREATE TABLE, CREATE VIEW TO korisnik;**
- **GRANT CREATE PROCEDURE TO korisnik;**

Za naredbe DML-a

Primeri:

- **GRANT SELECT ON student TO stuslu;**
- **GRANT SELECT(ime_stud, prez_stud) ON student TO korisnik;**
- **GRANT DELETE ON mesto TO korisnik;**

REVOKE

Oduzimanje prava korisnicima na izvršavanje SQL naredbi i/ili kreiranje bjekata (suprotno od naredbe GRANT)

Sintaksa za naredbe:

REVOKE { ALL | *statement* [,...*n*] } FROM *security_account* [,...*n*]

Sintaksa za objekte:

REVOKE [GRANT OPTION FOR]
{ ALL [PRIVILEGES] | *permission* [,...*n*] }
{
[(*column* [,...*n*])] ON { *table* | *view* }
| ON { *table* | *view* } [(*column* [,...*n*])]
| ON { *stored_procedure* | *extended_procedure* }
| ON { *user_defined_function* }
}
{ TO | FROM } *security_account* [,...*n*]
[CASCADE]
[AS { *group* | *role* }]

SQL

Veoma česta upotreba naredbe SQL-a je za pretraživanje podataka u tabelama. Najčešće se koristi sledeći oblik, tj. naredba ***Select***.

Select [All/Distinct] < lista atributa >
From < lista tabela >
Where [Order by] < uslov >;

- <*Lista atributa* > su atributi relacija koji se žele u rezultatu. Vidljivo je da *Select* odgovara operaciji projekcije u relacionoj algebri.
- <*Lista tabela*> su relacije koje se koriste u pretraživanju, pa je *From* ekvivalentan operaciji Kartezijevog proizvoda u relacionoj algebri.
- <*Uslov*> je predikat koji zadovoljavaju selektovane n-torke u rezultatu, pa je *Where* ekvivalentan operaciji selekcije u relacionoj algebri.

Navedeni tipični upitni blok SQL jezika je samo deo mogućnosti, a ekvivalentan je već opisanom opštem izrazu kojim se zadovoljava upit u relacionoj algebri:

$$\pi_{A_1, A_2 \ldots, A_m} (\sigma_{A_i \theta A_j} (r_1 X r_2 \ldots X r_n))$$

Osnovna struktura

SQL je baziran na skupovnim i relacionim operacijama sa određenim modifikacijama i poboljšanjima

Tipičan SQL upit:

SELECT *A1, A2, ..., An*
FROM *r1, r2, ..., rm*
WHERE *P*

- ***Ai*** predstavlja atribute
- ri predstavlja relacije
- ***P*** predstavlja predikat

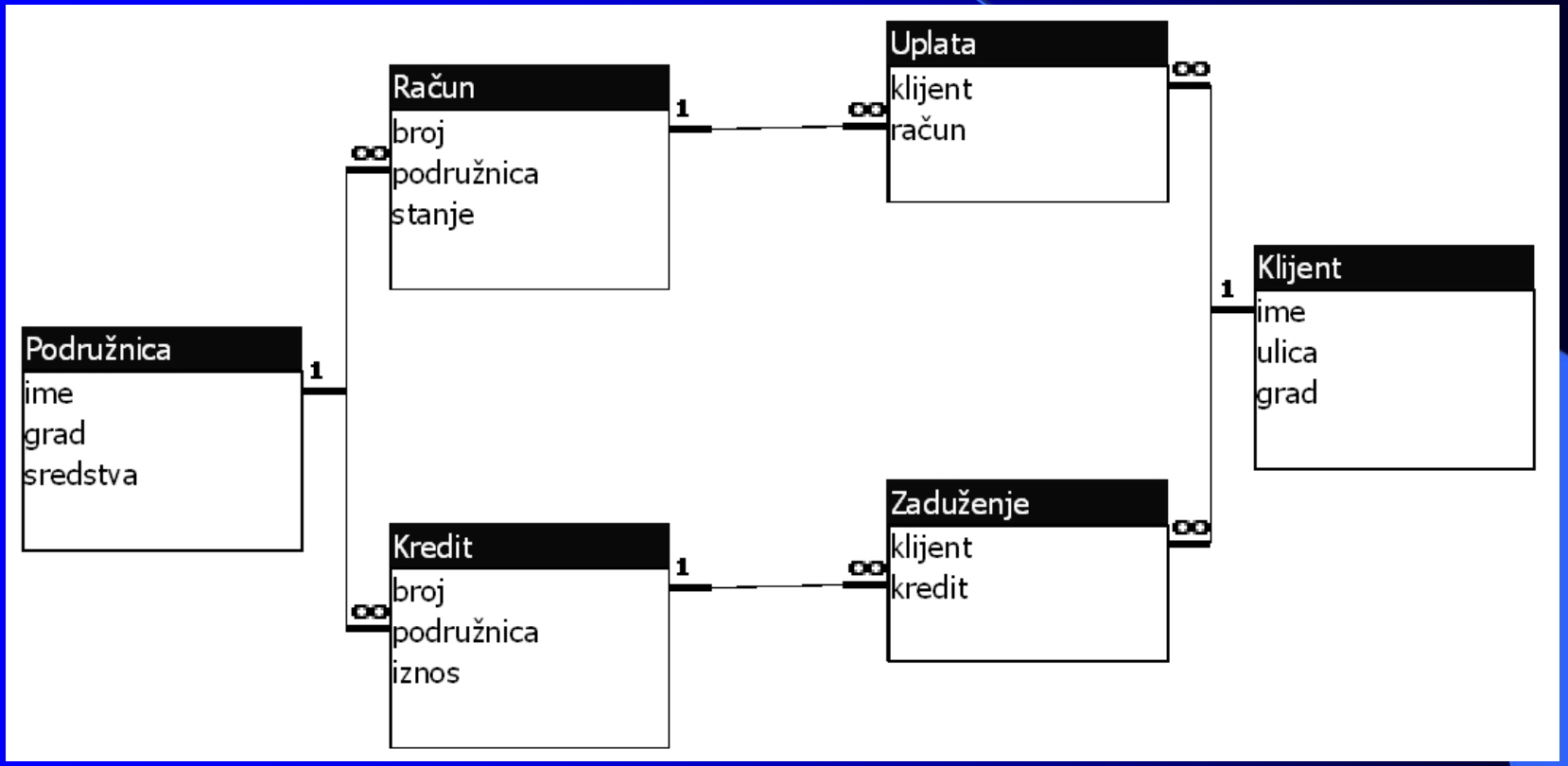
Primer je kao što znamo ekvivalentan izrazu u relacionoj algebri:

$$\Pi A1, A2, ..., An(\sigma P\ (r1 \times r2 \times ... \times rm))$$

Rezultat SQL upita je nova relacija.

PRIMER NAREDBE SELECT

Dat je deo IS svojim modelom:



Npr. pronađi imena svih podružnica u relaciji *kredit*

select *podružnica*

from *kredit*

- U sintaksi relacione algebre, upit je:

$\Pi_{podružnica}(kredit)$

- SQL ne razlikuje velika i mala slova u imenima

podružnica == PODRUŽNICA == PoDrUžNiCa

Naredba SELECT

- SQL dozvoljava duplikate u relacijama i rezultatima upita.
- Duplikate eliminšemo korišćenjem ključne reči **distinct** posle naredbe **select**

PRIMER: Pronađi imena svih podružnica u relaciji *kredit* i odbaci duplikate

select distinct *podružnica*
from *kredit*

Ključna reč **all** određuje da se duplikati NE eliminišu

select all *podružnica*
from *kredit*

Naredba SELECT

Zvezdica u **select** naredbi znači “sve atribute”

select *
from *kredit*

Naredba **select** može sadržavati aritmetičke izraze kao +, -, *, /, i operacije na konstantama ili atributima

Upit:

select *broj, podružnica, iznos* ∗ 100
from *kredit*

vratio bi relaciju gotovo jednaku relaciji *kredit*, sa razlikom *iznosa* koji je pomnožen sa 100

Neka tabela, na primer, mogla bi da sadrži jedinstveni matični broj građana, imena, prezimena i adrese zaposlenih:

TabelaAdresaRadnika					
JMBG	**Ime**	**Prezime**	**Adresa**	**Grad**	**Republika**
512687458	Đorđe	Petrović	Kralja Petra 9	Beograd	Srbija
758420012	Marija	Simić	Bul Nikole Tesle 22	Jagodina	Srbija
102254896	Savo	Jovanović	Njegoševa 17	Podgorica	Crna Gora
876512563	Svetlana	Aćimović	Laze Lazarevića 10	Subotica	Srbija

- Pretpostavimo da želimo, recimo, da vidimo adrese svih zaposlenih. Da bi to postigli, koristimo naredbu SELECT:

```
SELECT Ime, Prezime, Adresa, Grad, Republika
FROM TabelaAdresaRadnika;
```

- Rezultat ovog *upita* u bazu podataka je:

Ime	Prezime	Adresa	Grad	Republika
Đorđe	Petrović	Kralja Petra 9	Beograd	Srbija
Marija	Simić	Bul Nikole Tesle 22	Jagodina	Srbija
Savo	Jovanović	Njegoševa 17	Podgorica	Crna Gora
Svetlana	Aćimović	Laze Lazarevića 10	Subotica	Srbija

Neka je data relacija PROIZVOD sa atributima:
ŠifraP, NazivP, Jed.M., Količina, Cena

relacija Proizvod

ŠifraP	*Naziv P*	*Jed.M.*	*Količina*	*Cena*
342	Hleb	komad	32	40
456	Pivo	litar	452	60
122	Cigarete	komad	45	100
121	Cigarete	komad	523	70
768	Ulje	litra	34	80

PRIMER za tabelu PROIZVOD :

- Upotrebom naredbe *Select* prikazati nazive proizvoda koji se nalaze na zalihi.

Select NazivP
From *Proizvod*;

NazivP
Hleb
Pivo
Cigarete
Cigarete
Ulje

Rezultat nije prava projekcija jer postoje n-torke koje se ponavljaju. Ako se želi prava projekcija, onda upit ima sledeći oblik:

Select Distinct NazivP
FROM Proizvod;

NazivP
Hleb
Pivo
Cigarete
Ulje

Relacioni operatori

=	Jednako
!=	Različito
<	Manje
>	Veće
<=	Manje ili jednako
>=	Veće ili jednako

U SQL-u postoji šest relacionih operatora i posle njihovog predstavljanja videćemo kako se koriste.

Da bi se prikazali samo oni redovi iz određene tabele koji zadovoljavaju postavljene kriterijume, koristi se *klauzula WHERE*. Ona se može najlakše razumeti ukoliko se pogleda nekoliko primera.

Naredba WHERE

Naredba **where** određuje uslove koje rezultat mora zadovoljavati i odgovara predikatu selekcije u relacionoj algebri

Primer: Pronađi sve brojeve računa kredita podignutih u podružnici Niš na iznose veće od 12,000 rsd.

select *broj*
from *kredit*
where *podružnica* = "Niš" and *iznos* > 12000

- Uslovi predikata mogu se kombinovati korišćenjem logičkih operatora **and, or i not** (konjunkcija, disjunkcija i negacija)
- Poređenja se mogu raditi i sa rezultatima aritmetičkih operacija

Naredba WHERE

TabelaPrimanjaRadnika			
IDRadnika	**Plata**	**Prinadležnosti**	**Položaj**
010	75000	15000	rukovodilac
105	65000	15000	rukovodilac
152	60000	15000	rukovodilac
215	60000	12500	rukovodilac
244	50000	12000	činovnik
300	45000	10000	činovnik
335	40000	10000	činovnik
400	32000	7500	pripravnik
441	28000	7500	pripravnik

Ukoliko želimo da dobijemo ID brojeve onih zaposlenih koji zarađuju preko 50.000, koristćemo sledeću naredbu. Opis klauzule WHERE, odnosno deo PLATA >= 50000, naziva se *uslov* (operacija koja kao rezultat daje vrednost True (tačno) ili False (netačno)).

```
SELECT IDRADNIKA
FROM TABELAPRIMANJARADNIKA
WHERE PLATA >= 50000;
```

Naredba WHERE

TabelaPrimanjaRadnika			
IDRadnika	**Plata**	**Prinadležnosti**	**Položaj**
010	75000	15000	rukovodilac
105	65000	15000	rukovodilac
152	60000	15000	rukovodilac
215	60000	12500	rukovodilac
244	50000	12000	činovnik
300	45000	10000	činovnik
335	40000	10000	činovnik
400	32000	7500	pripravnik
441	28000	7500	pripravnik

Sledeća naredba prikazuje ID brojeve svih rukovodilaca:

```
SELECT IDRADNIKA
FROM
TABELAPRIMANJARADNIKA
WHERE POLOŽAJ = 'rukovodilac';
```

Na primer, da bi prikazali sve činovnike koji zarađuju više od 40.000, koristimo naredbu:

```
SELECT IDRADNIKA
FROM TABELAPRIMANJARADNIKA
WHERE PLATA > 40000 AND POLOŽAJ = 'činovnik';
```

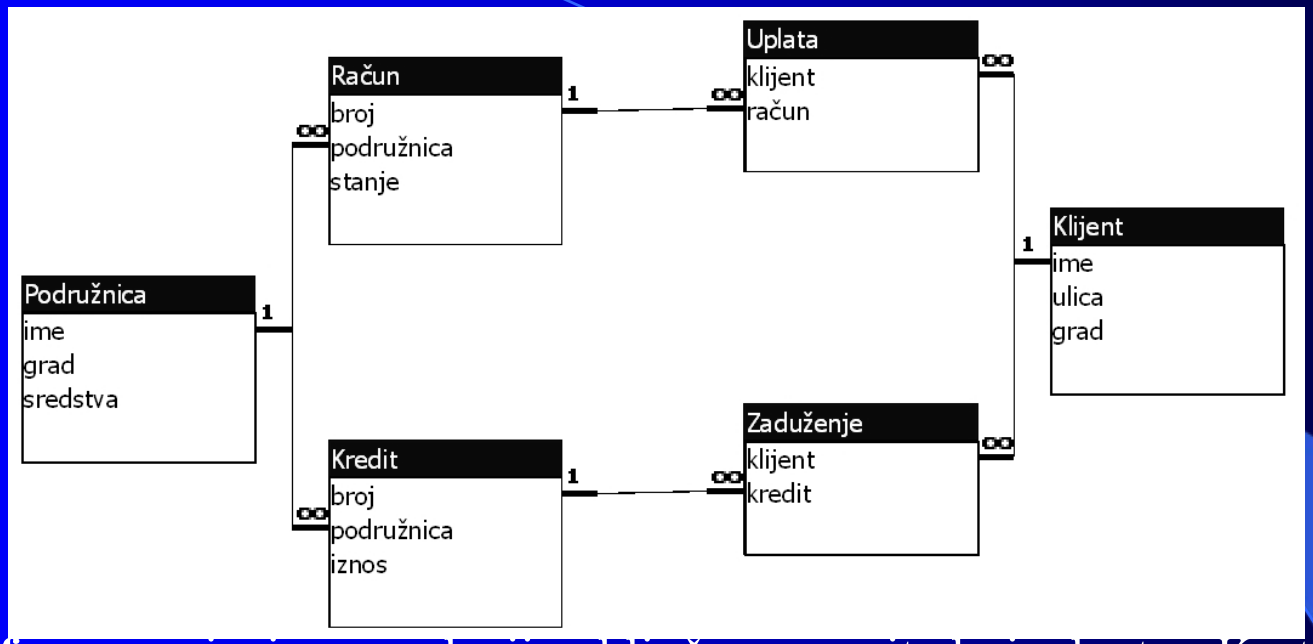
Naredba WHERE

TabelaPrimanjaRadnika			
IDRadnika	**Plata**	**Prinadležnosti**	**Položaj**
010	75000	15000	rukovodilac
105	65000	15000	rukovodilac
152	60000	15000	rukovodilac
215	60000	12500	rukovodilac
244	50000	12000	činovnik
300	45000	10000	činovnik
335	40000	10000	činovnik
400	32000	7500	pripravnik
441	28000	7500	pripravnik

Naći redove u kojima zaposleni ima platu veću od 60.000 i ima rukovodeći položaj, a zatim iz ove liste redova izdvojiti one koji zadovoljavaju uslov da su prinadležnosti veće od 12.000.

```
SELECT IDRADNIKA
FROM TABELAPRIMANJARADNIKA
WHERE POLOŽAJ = 'rukovodilac' AND PLATA > 60000 OR
PRINADLEŽNOSTI > 12000;
```

Naredba WHERE FROM n tabela



Naredba **from** popisuje sve relacije uključene u upit ekvivalentno Kartezijevom proizvodu u relacionoj algebri i može biti vezana za više tabela

PRIMER: Pronađi Kartezijev proizvod relacija *kredita* i *zaduženja:*

select ∗
from Zaduženje, Kredit

PRIMER: Pronađi ime, broj računa kredita i iznos kredita svih klijenata koji su se zadužili u podružnici Niš:

select *klijent, kredit, iznos* (*kredit* ili *broj*)
from Zaduženje, Kredit
where Zaduženje.*kredit* = Kredit.*broj* and *podružnica* = “Niš”

Naredba WHERE FROM n tabela

Primer: Pronađi imena svih klijenata i brojeve računa njihovih kredita za sve klijente koji su uzeli kredit

select *klijent, Z.kredit, K.iznos*
from *zaduženje* as *Z, kredit* as *K*
where *Z.kredit = K.broj*

Primjer: Pronađi imena svih podružnica koje imaju više sredstava od podružnice od svih koje se nalaze u Nišu

select distinct *P.ime*
from *podružnica* as *P, podružnica* as *P2*
where *P.sredstva* > *P2.sredstva* and *P2.grad* =“Niš”

Promena imena

- SQL omogućuje promenu imena relacija i atributa
- koriš'enjem naredbe **as**

staro_ime **as** *novo_ime*

- Primer: Pronađi imena, brojeve računa kredita i iznose kredita svih klijenata; promeni ime atributa *kredit* u *broj_kredita*

select *klijent, zaduženje.kredit* **as** *broj_kredita, iznos*
from *zaduženje, kredit*
where *zaduženje.kredit = kredit.broj*

Promena imena

- Primer: Pronađi imena svih klijenata i brojeve računa njihovih kredita za sve klijente koji su uzeli kredit

select *klijent, Z.kredit, K.iznos*
from *zaduženje* **as** *Z, kredit* **as** *K*
where *Z.kredit = K.broj*

- Primer: Pronađi imena svih podružnica koje imaju više
- sredstava od podružnice od svih koje se nalaze u Zemunu

select distinct *P.ime*
from *podružnica* **as** *P, podružnica* **as** *P2*
where *P.sredstva > P2.sredstva* **and** *P2.grad =*
"Zemun"

Operacije na stringovima

- SQL ima operator za poredjenje nizova znakova. Uzorci su opisani posebnim znakovima:

***** označava bilo koji podniz
? označava bilo koji znak

- Primjer: Pronađi imena svih klijenata koji žive u ulicama bez broja

select *ime*
from *klijent*
where *ulica* **like** “**b.b.”

Neke od dozvoljenih operacija nad nizovima znakova

- spajanje nizova znakova (operator **&**)
- promena velika-mala slova
- izračunavanje duzine niza znakova

Redosled

- Primer: Pronađi imena svih klijenata koji imaju kredit u podružnici Zemun-Centar

select distinct *klijent*
from *zaduženje, kredit*
where *zaduženje.kredit* = *kredit.broj* **and**
podružnica = "Rijeka-Centar"
order by *klijent*

- Dodavanjem ključne reči **desc** redosled će biti silazni, dok će za **asc** biti uzlazni (**asc** je pretpostavljen)

- **order by** *klijent* **desc**

Duplikati

- Za relacije sa duplikatima SQL može odrediti koliko će kopija objekata biti u rezultatu
- Više-vrednosne (eng. *multiset*) verzije nekih operatora relacione algebre za više-vrednosne relacije *r*1 i *r*2:

1. σθ **(r1)** – Ako postoji *c*1 kopija objekta *t*1 u *r*1, i *t*1 zadovoljava odabir σθ,, onda postoji *c*1 kopija objekta *t*1 u σθ *(r*1*)*

2. Π**A(r1)** – Za svaku kopiju objekta *t1* u *r*1, postoji kopija objekta Π*A(t*1*)* u Π*A(r*1*)* gdje Π*A(t*1*)* predstavlja projekciju samog objekta *t1*

3. r1 x r2 – Ako postoji *c*1 kopija objekta *t1* u *r*1 i *c*2 kopija objekta *t*2 u *r*2, postoji *c*1 x *c*2 kopija objekta *t1*. *t*2 u *r*1 x *r*2

Duplikati

- Primer: Neka su relacije $r1$ (A, B) i $r2$ (C) popunjene ovako:
- $r1 = \{(1, a)\ (2,a)\}$ $r2 = \{(2), (3), (3)\}$
- $\Pi B(r1) = \{(a), (a)\}$
- $\Pi B(r1)$ x $r2 = \{(a, 2), (a, 2), (a, 3), (a, 3), (a, 3), (a, 3)\}$
- SQL semantika duplikata

select $A1,, A2, ..., An$
from $r_1, r_2, ..., r_m$
where P

- ekvivalentna je više-vrednosnoj verziji izraza
  $\Pi\ A1,, A2, ..., An(\sigma P\ (r_1$ x $r_2\ x\ ...\ x\ r_m))$

Operacije nad skupovima

Podržane su operacije nad skupovima:

- **union** – unija U
- **intersect** – presek ∩
- **except** – razlika /
- Svaka od navedenih operacija automatski ukloni duplikate; duplikati se mogu zadržati koriš'enjem ključne reči **all**

- Npr. objekt se pojavi *m* puta u *r1* i *n* puta u *r2*:

- *r1* **union all** r2 m+n
- *r1* **intersect all** *r2* min(m,n)
- *r1* **except all** *r2* max(0, *m – n)*

Operacije na skupovima

- Pronađi sve klijente koji imaju kredit, račun ili oboje

  (select *klijent* **from** *uplata*)
  union
  (select *klijent* **from** *zaduženje)*

- Pronađi sve klijente koji imaju i kredit i račun

  (select *klijent* **from** *uplata*)
  intersect
  (select *klijent* **from** *zaduženje)*

- Pronađi sve klijente koji imaju račun, ali nemaju kredit

  (select *klijent* **from** *uplata*)
  except
  (select *klijent* **from** *zaduženje)*

Agregatne funkcije

- Agregatne funkcije izvode se na podskupu objekata atributa relacije i vraćaju neku vrednost

- **avg** srednja vrednost
- **min** najmanja vrednost
- **max** najveća vrednost
- **sum** suma vrednosti
- **count** broj vrednosti

Agregatne funkcije

- Pronađi srednji iznos stanja (novca) u podružnici Zemun-Centar

  select avg *(stanje)*
  from *račun*
  where *podružnica* = “Zemun-Centar”

- Pronađi broj objekata u relaciji klijent

  select count (*)
  from *klijent*

- Pronađi broj klijenata koji su izvršili barem jednu uplatu

  select count (distinct *klijent)*
  from *uplata*

Agregatne funkcije – group by

- Pronađi broj uplatitelja u svakoj podružnici

select *podružnica,* **count (distinct** *klijent)*
from *uplata, račun*
where *uplata.račun = račun.broj*
group by *podružnica*

Agregatne funkcije – having

- Pronađi imena svih podružnica gde je prosečno stanje na računima više od 1,200 rsd

select *podružnica*, **avg** *(stanje)*
from *račun*
group by *podružnica*
having avg *(stanje)* > 1200

- Predikati u izrazu **having** dolaze posle grupisanja, dok se predikati u **where** izrazu primenjuju pre grupisanja

Ugnježđivanje

- SQL podržava ugnježđivanje – upite unutar upita
- Podupit je izraz tipa **select-from-where** koji se nalazi unutar drugog upita
- Obično se koriste za provjeru članstva u skupu, usporedbu skupova i izračun kardinalnosti skupova

Ugnježđivanje

- Pronađi sve klijente koji imaju i račun i kredit u banci

select distinct *klijent*
from *zaduženje*
where *klijent* **in**
(select *klijent* **from** uplata)

- Pronađi sve klijente koji u banci imaju kredit ali nemaju račun

select distinct *klijent*
from *zaduženje*
where *klijent* **not in**
(select *klijent* **from** *uplata)*

Ugnježđivanje

- Pronađi sve klijent koji imaju i račun i kredit u podružnici Zemun-Centar

select distinct *klijent*
from *zaduženje, kredit*
where *zaduženje.kredit = kredit.broj* **and**
podružnica = "Zemun -Centar" **and**
(podružnica, klijent) **in**
(select *podružnica, klijent*
from *uplata, račun*
where *uplata.račun = račun.broj)*

- Napomena: Ovaj se izraz može napisati puno jednostavnije

Poredjenje skupova

- Pronađi sve podružnice koje imaju više sredstava od bilo koje podružnice smještene u Zemunu

select distinct *P.ime*
from *podružnica* **as** *P, podružnica* **as** *P2*
where *P.sredstva* > *P2.sredstva* **and** *P2.grad* = *"Zemun"*

- Isti upit korištenjem izraza **> some**

select *ime*
from *podružnica*
where *sredstva* > **some**
(select *sredstva* **from** *podružnica*
where *grad* = *"Zemun"*)

Prazne relacije

- Izraz **exists** vraća vrijednost istina ako relacija nije prazna

- **exists** r □ $r \neq \emptyset$
- **not exists** $r \Leftrightarrow r = \emptyset$

Prazne relacije

- Pronađi sve klijente koji imaju račun u svim podružnicama koje se nalaze u Zemunu

select distinct *U.klijent*
from *uplata* **as** *U*
where not exists (
(select *ime*
from *podružnica*
where *grad* =“Zemun”)
except
(select *R.podružnica*
from *uplata* **as** *U2, račun* **as** *R*
where *U2.račun = R.broj* **and**
U.klijent = U2.klijent))

Operacija join

- Operacija **join** uzima dvije relacije i kao rezultat vraća treću

- Najčešće se koristi kao podupit u izrazu **from**
- Tip – određuje na koji se način tretiraju podskupovi objekata u dvoma relacijama koji serazlikuju:

- **inner join**
- **left outer join**
- **right outer join**

Operacija join

- Relacija *kredit*

broj	*podružnica*	iznos
L-170	Rijeka-Centar	3000
L-230	Rijeka-Vežica	4000
L-260	Opatija	1700

- Relacija *zaduženje*

klijent	*kredit*
Ana	L-170
Marin	L-230
Danko	L-155

Napomena:

Nedostaju informacije o zaduženju L-260 i kreditu L-155

Operacija join

- **select** * **from** *kredit* **inner join** *zaduženje*
  on *kredit.broj* = *zaduženje.kredit*

broj	*podružnica*	iznos	*klijent*	*kredit*
L-170	Rijeka-Centar	3000	Ana	L-170
L-230	Rijeka-Vežica	4000	Marin	L-230

broj	*podružnica*	iznos
L-170	Rijeka-Centar	3000
L-230	Rijeka-Vežica	4000
L-260	Opatija	1700

klijent	*kredit*
Ana	L-170
Marin	L-230
Danko	L-155

Operacija left outer join

- select * from *kredit* left join *zaduženje*
  on *kredit.broj* = *zaduženje.kredit*

broj	*podružnica*	iznos	*klijent*	*kredit*
L-170	Rijeka-Centar	3000	Ana	L-170
L-230	Rijeka-Vežica	4000	Marin	L-230
L-260	Opatija	1700	*null*	*null*

broj	*podružnica*	iznos
L-170	Rijeka-Centar	3000
L-230	Rijeka-Vežica	4000
L-260	Opatija	1700

klijent	*kredit*
Ana	L-170
Marin	L-230
Danko	L-155

Operacija right outer join

- select * from *kredit* right join *zaduženje* on *kredit.broj* = *zaduženje.kredit*

broj	podružnica	iznos	klijent	kredit
L-170	Rijeka-Centar	3000	Ana	L-170
L-230	Rijeka-Vežica	4000	Marin	L-230
null	*null*	*null*	Danko	L-155

broj	podružnica	iznos
L-170	Rijeka-Centar	3000
L-230	Rijeka-Vežica	4000
L-260	Opatija	1700

klijent	kredit
Ana	L-170
Marin	L-230
Danko	L-155

Brisanje zapisa - delete

- Pobriši sve zapise o računima u podružnici Zemun-Centar

delete from *račun*
where *podružnica* = “Zemun -Centar”

- Pobriši sve račune u svim podružnicama u gradu Zemunu

delete from *račun*
where *podružnica* **in**
(**select** *ime* **from** *podružnica*
where *grad* = “Zemun”)

Brisanje zapisa - delete

- Pobriši sve zapise računa kojima je stanje ispod proseka u banci

delete from *račun*
where *stanje* < (**select avg** *(stanje)* **from** *račun)*

- Problem: kako se brišu zapisi, tako se menja srednje stanje na računima u banci
- Rešenje:

1. Izračunamo srednju vrednost stanja računa i nađemo zapise za brisanje
2. Pobrišemo sve zapise predviđene za brisanje

Dodavanje zapisa - insert

- Dodaj novi zapis u *račun*

  insert into *račun*
  values (“A-9732”, “Zemun”, 1200)

- ili

  insert into *račun (podružnica, stanje, broj)*
  values (“Zemun”, 1200, “A-9732”)

  Dodaj novi zapis u *račun*, gde je *stanje* nul vrednost

  insert into *račun*
  values (“A-777”, ”Zemun”, *null*)

Dodavanje zapisa - insert

- Svim klijentima koji u podružnici Zemun imaju kredit, kao poklon otvori štedne račune sa početnih 200 rsd. Neka broj kredita služi kao broj računa štednje koji se otvara.

insert into *račun*
select *broj, podružnica,* 200
from *kredit*
where *podružnica* = “Zemun”
insert into *uplata*
select *klijent, broj*
from *kredit, zaduženje*
where *podružnica* = “Zemun”
and *kredit.broj = zaduženje.kredit*

Promena zapisa - update

- Povećaj stanje svih računa s više od 10,000 rsd za %, a svih ostalih za 5 %

update *račun*
set *stanje* = *stanje* ∗ 1.06
where *stanje* > 10000
update *račun*
set *stanje* = *stanje* ∗ 1.05
where *stanje* ≤ 10000

- Redosled je važan!!!!!