



IT250 - BAZE PODATAKA

SQL: Naredba SELECT za rad sa
više tabela; Kreiranje pogleda

Lekcija 10

PRIRUČNIK ZA STUDENTE

IT250 - BAZE PODATAKA

Lekcija 10

SQL: NAREDBA SELECT ZA RAD SA VIŠE TABELA; KREIRANJE POGLEDA

- ✓ SQL: Naredba SELECT za rad sa više tabela; Kreiranje pogleda
- ✓ Poglavlje 1: Spajanje tabela korišćenjem podupita
- ✓ Poglavlje 2: JOIN za spajanje tabela
- ✓ Poglavlje 3: LEFT (OUTER) JOIN
- ✓ Poglavlje 4: RIGHT (OUTER) JOIN
- ✓ Poglavlje 5: FULL (OUTER) JOIN
- ✓ Poglavlje 6: SELF JOIN
- ✓ Poglavlje 7: Korišćenje pogleda (eng. VIEW)
- ✓ Poglavlje 8: Pokazna vežba
- ✓ Poglavlje 9: Domaći zadatak
- ✓ Zaključak

Copyright © 2017 – UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

▼ Uvod

UVOD

Šta ćemo naučiti u ovoj lekciji?

Naredbom SELECT se mogu dobiti sadržaji i iz više tabela U. za šta se mogu primeniti dve metode:

1. spajanje (JOIN) tabela i
2. korišćenje podupita.

U predavanju su dati primeri korišćenja oba ova načina i ukazano je na njihove prednosti i nedostatke.

Obzirom na ograničenja koja postoje prilikom korišćenja podupita, mnogo češći način za dobijanje podataka iz više tabela U. je spajanje (JOIN).

U INNER JOIN, koji se može smatrati podrazumevanim (eng.**default**) tipom spajanja, torka jedne relacije je uključena u rezultat samo ako se podudara sa torkom u drugoj relaciji prema uslovu spajanja.

Ako korisnik zahteva da budu uključene sve torke relacije u JOIN-u a ne samo one koje se podudaraju sa torkama u drugoj relaciji prema uslovu spajanja kao što je to slučaj kod INNER JOIN, eksplicitno mora biti korišćen OUTER JOIN:

OUTER JOIN se može javiti kao:

1.

LEFT OUTER JOIN: u rezultatu mora da se pojavi svaka torka iz leve relacije, koja može biti ispunjena NULL vrednostima za attribute desne relacije,

2.

RIGHT OUTER JOIN: u rezultatu mora da se pojavi svaka torka iz desne relacije, ako nema odgovarajuće torke, ispunjena je NULL vrednostima za attribute leve relacije,

3.

FULL OUTER JOIN: u rezultatu se pojavljuju sve torke iz leve relacije koje mogu biti ispunjene NULL vrednostima za attribute desne relacije kao i sve torke iz desne relacije koje mogu biti ispunjene NULL vrednostima za attribute leve relacije

U drugom delu predavanja se govori o pogledima (VIEW) koji predstavljaju virtuelne tabele koje se kreiraju iz drugih tabela U. ili **view**-ova.

Sa njim se radi gotovo isto kao i sa baznom tabelom, mada view nema svoje podatke i ne zauzima nikakav memorijski prostor.

▼ Poglavlje 1

Spajanje tabela korišćenjem podupita

TABELA RJ ZA PRAVLJENJE PODUPITA ZA SPAJANJE TABELA

Struktura tabele RJ

SQL raspolaže sa dve različite tehnike za pravljenje upita nad više tabela:

1. korišćenje podupita tj. umetanje upita nad jednom relacijom u upit nad drugom i
2. JOIN operacije za kombinovanje više tabela.

Mada se obe tehnike koriste za rad sa više tabela, one se koriste u različitim situacijama.

U ovom segmentu će biti reči o pravljenju upita nad više tabela umetanjem upita nad jednom relacijom u upit nad drugom. To će biti ilustrovano primerima upita na primeru relacionog modela sastavljenog od dve relacije koje su predstavljene na ovoj i narednoj sekciji:

1. R_J (RADNA_JEDINICA) i
2. RADNIK

Relacija (Tabela) R_J (SQL kod):

```
CREATE TABLE `it350l06`.`R_J` (  
  `S_RJ` INT NOT NULL ,  
  `NAZIV` VARCHAR(45) NULL ,  
  `GRAD` VARCHAR(45) NULL ,  
  PRIMARY KEY (`S_RJ`) )  
ENGINE = InnoDB  
DEFAULT CHARACTER SET = utf8;
```

Odgovarajući prikaz tabele i podatka u tabeli, kao i unos podataka u SQL-u za datu tabelu:

	S_RJ	NAZIV	GRAD
	10	PRODAJA	BEOGRAD
	20	PROIZVODNJA	NIS
	30	PROJEKTOVA...	BOR
..	40	ISTRAZIVANJE	NOVI SAD

Slika 1.1 Popunjena tabela R_J (RADNA JEDINICA) [Izvor: Autor]

```
INSERT INTO `it350l06`.`R_J` (`S_RJ`, `NAZIV`, `GRAD`)
VALUES (10, 'PRODAJA', 'BEOGRAD');
INSERT INTO `it350l06`.`R_J` (`S_RJ`, `NAZIV`, `GRAD`)
VALUES (20, 'PROIZVODNJA', 'NIS');
INSERT INTO `it350l06`.`R_J` (`S_RJ`, `NAZIV`, `GRAD`)
VALUES (30, 'PROJEKTOVANJE', 'BOR');
INSERT INTO `it350l06`.`R_J` (`S_RJ`, `NAZIV`, `GRAD`)
VALUES (40, 'ISTRAZIVANJE', 'NOVI SAD');
```

TABELA RADNIK ZA PRAVLJENJE PODUPITA ZA SPAJANJE TABELA

Struktura tabele RADNIK

Tabela RADNIK (SQL kod), odgovarajući prikaz tabele i podatka u tabeli, kao i unos podataka u SQL-u za datu tabelu

```
CREATE TABLE `it350l06`.`RADNIK` (
  `S_RADNIK` INT NOT NULL ,
  `S_RJ` INT NULL ,
  `IME` VARCHAR(45) NULL ,
  `POSAO` VARCHAR(45) NULL ,
  `S_RUKO` INT NULL ,
  `DAT_ZAP` DATE NULL ,
  `LD` INT(13) NULL ,
  `PREMIJA` INT(13) NULL ,
  PRIMARY KEY (`S_RADNIK`) ,
  INDEX `S_RJ` (`S_RJ` ASC) ,
  CONSTRAINT `S_RJ`
    FOREIGN KEY (`S_RJ`)
    REFERENCES `it350l06`.`r_j` (`S_RJ`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
```

	S_RADNIK	S_RJ	IME	POSAO	S_RUKO	DAT_ZAP	LD	PREMIJA
▶	3069	20	STEFAN	ANALITICAR	3602	2000-12-17	80000	NULL
	3199	30	MILAN	TRG PUTNIK	3398	3001-02-20	160000	30000
	3221	30	PETAR	TRG PUTNIK	3398	2001-02-21	125000	50000
	3266	20	MARKO	UKOVODILAC	3539	2001-04-02	297500	NULL
	3354	30	MARIJA	TRG PUTNIK	3398	2001-09-01	125000	140000
	3398	30	ANA	UKOVODILAC	3539	2001-05-01	285000	NULL
	3482	10	IVAN	UKOVODILAC	3539	2001-06-09	245000	NULL
	3483	20	PAVLE	SAVETNIK	3266	2001-11-09	300000	NULL
	3539	10	JOVAN	PREDSEDIK	NULL	2001-11-17	500000	NULL
	3544	30	GORAN	TRG PUTNIK	3398	2001-09-08	150000	0
	3579	20	JELENA	ANALITICAR	3488	2001-09-23	110000	NULL
	3600	30	JANKO	ANALITICAR	3398	2001-12-03	95000	NULL
	3602	10	FILIP	SAVETNIK	3266	2001-12-03	300000	NULL
	3634	10	DEJAN	ANALITICAR	34882	2002-01-23	130000	NULL

Slika 1.2 Popunjena tabela RADNIK [Izvor: NM IT250-2020/2021.]

```

INSERT INTO `it350l06`.`RADNIK` (`S_RADNIK`, `S_RJ`, `IME`, `POSAO`, `S_RUKO`,
`DAT_ZAP`, `LD`)
VALUES (3069, 20, 'STEFAN', 'ANALITICAR', 3602, '2000-12-17', 80000);

INSERT INTO `it350l06`.`RADNIK` (`S_RADNIK`, `S_RJ`, `IME`, `POSAO`, `S_RUKO`,
`DAT_ZAP`, `LD`, `PREMIJA`)
VALUES (3199, 30, 'MILAN', 'TRG PUTNIK', 3398, '3001-02-20', 160000, 30000);

INSERT INTO `it350l06`.`RADNIK` (`S_RADNIK`, `S_RJ`, `IME`, `POSAO`, `S_RUKO`,
`DAT_ZAP`, `LD`, `PREMIJA`)
VALUES (3221, 30, 'PETAR', 'TRG PUTNIK', 3398, '2001-02-21', 125000, 50000);

INSERT INTO `it350l06`.`RADNIK` (`S_RADNIK`, `S_RJ`, `IME`, `POSAO`, `S_RUKO`,
`DAT_ZAP`, `LD`)
VALUES (3266, 20, 'MARKO', 'UKOVODILAC', 3539, '2001-04-02', 297500);

INSERT INTO `it350l06`.`RADNIK` (`S_RADNIK`, `S_RJ`, `IME`, `POSAO`, `S_RUKO`,
`DAT_ZAP`, `LD`, `PREMIJA`)
VALUES (3354, 30, 'MARIJA', 'TRG PUTNIK', 3398, '2001-09-01', 125000, 140000);

INSERT INTO `it350l06`.`RADNIK` (`S_RADNIK`, `S_RJ`, `IME`, `POSAO`, `S_RUKO`,
`DAT_ZAP`, `LD`)
VALUES (3398, 30, 'ANA', 'UKOVODILAC', 3539, '2001-05-01', 285000);

INSERT INTO `it350l06`.`RADNIK` (`S_RADNIK`, `S_RJ`, `IME`, `POSAO`, `S_RUKO`,
`DAT_ZAP`, `LD`)
VALUES (3482, 10, 'IVAN', 'UKOVODILAC', 3539, '2001-06-09', 245000);

INSERT INTO `it350l06`.`RADNIK` (`S_RADNIK`, `S_RJ`, `IME`, `POSAO`, `S_RUKO`,
`DAT_ZAP`, `LD`)
VALUES (3483, 20, 'PAVLE', 'SAVETNIK', 3266, '2001-11-09', 300000);

INSERT INTO `it350l06`.`RADNIK` (`S_RADNIK`, `S_RJ`, `IME`, `POSAO`, `DAT_ZAP`,
`LD`)
VALUES (3539, 10, 'JOVAN', 'PREDSEDIK', '2001-11-17', 500000);

```

```
INSERT INTO `it350l06`.`RADNIK` (`S_RADNIK`, `S_RJ`, `IME`, `POSAO`, `S_RUKO`,  
`DAT_ZAP`, `LD`, `PREMIJA`)  
VALUES (3544, 30, 'GORAN', 'TRG PUTNIK', 3398, '2001-09-08', 150000, 0);  
  
INSERT INTO `it350l06`.`RADNIK` (`S_RADNIK`, `S_RJ`, `IME`, `POSAO`, `S_RUKO`,  
`DAT_ZAP`, `LD`)  
VALUES (3579, 20, 'JELENA', 'ANALITICAR', 3488, '2001-09-23', 110000);  
  
INSERT INTO `it350l06`.`RADNIK` (`S_RADNIK`, `S_RJ`, `IME`, `POSAO`, `S_RUKO`,  
`DAT_ZAP`, `LD`)  
VALUES (3600, 30, 'JANKO', 'ANALITICAR', 3398, '2001-12-03', 95000);  
  
INSERT INTO `it350l06`.`RADNIK` (`S_RADNIK`, `S_RJ`, `IME`, `POSAO`, `S_RUKO`,  
`DAT_ZAP`, `LD`)  
VALUES (3634, 10, 'DEJAN', 'ANALITICAR', 34882, '2002-01-23', 130000);  
  
INSERT INTO `it350l06`.`RADNIK` (`S_RADNIK`, `S_RJ`, `IME`, `POSAO`, `S_RUKO`,  
`DAT_ZAP`, `LD`)  
VALUES (3602, 10, 'FILIP', 'SAVETNIK', 3266, '2001-12-03', 300000);
```

PODUPIT ZA SPAJANJE TABELA

Primer u kojem se može upotrebiti podupit za spajanje tabela

Jedan od načina povezivanja tabela relacione baze podataka je korišćenje podupita za spajanje tabela, preciznije rečeno, dinamička zamena rezultata jednog upita u WHERE klauzuli drugog.

Primer koji ukazuje na situaciju kada se mogu spojiti dva upita je sledeći: **Prikazati ime i posao svakog radnika koji ima isti posao kao Dejan.**

Najpre bi trebalo utvrditi koji posao obavlja Dejan. To se realizuje sledećom SQL naredbom

```
SELECT POSAO  
FROM RADNIK  
WHERE IME = 'DEJAN';
```

Iz ove naredbe saznajemo da DEJAN obavlja posao analitičara (slika 1.3.)

	POSAO
▶	ANALITICAR

Slika 1.3 Rezultat SELECT naredbe [Izvor: Autor]

Sada se polazni zahtev svodi na prikazivanje imena i posla svakog radnika koji obavlja posao analitičara, što se realizuje sledećom naredbom.

```
SELECT IME, POSAO  
FROM RADNIK  
WHERE POSAO = 'ANALITICAR';
```

Rezultati ove naredbe dati su na slici 1.4:

	IME	POSAD
▶	STEFAN	ANALITICAR
	JELENA	ANALITICAR
	JANKO	ANALITICAR
	DEJAN	ANALITICAR

Slika 1.4 Rezultat za SELECT naredbu [Izvor: Autor]

Kad god se uslov u WHERE klauzuli ugnježdjenog upita referencira na neki atribut relacije deklarisan u spoljnom upitu, za dva upita se može reći da su povezana. **Podupit za spajanje tabela možemo bolje da razumemo uzimajući u obzir da se ugnježdjeni upit (onaj koji se nalazi u WHERE klauzuli) proverava po jednom za svaku torku (ili kombinaciju torki) u spoljnom upitu.**

KAKO POVEZATI PODUPIT I UPIT

Primer povezivanja podupita i upita za spajanje tabela

Povezivanje tabela dinamičkom zamenom rezultata jednog upita u WHERE klauzuli drugog, se sastoji upravo u tome da se umesto rezultata prvog upita (u konkretnom slučaju 'ANALITIČAR') u WHERE klauzuli drugog, piše prvi upit tj. upit koji vraća taj rezultat (u konkretnom slučaju), tako da SELECT naredba ima sledeći izgled

```
SELECT IME, POSAO  
FROM RADNIK  
WHERE POSAO = (  
    SELECT POSAO  
    FROM RADNIK  
    WHERE IME = 'DEJAN'  
);
```

Dobijeni rezultat (*slika 1.5*) je isti kao kada smo izvršili upite iz prethodnog primera, sa tom razlikom da smo sada to radili jednim upitom.

	IME	POSLO
▶	STEFAN	ANALITICAR
	JELENA	ANALITICAR
	JANKO	ANALITICAR
	DEJAN	ANALITICAR

Slika 1.5 Rezultat SELECT naredbe [Izvor: Autor]

Prvi upit, odnosno upit u zagradama se naziva unutrašnji upit i on se uvek izvršava prvi. Drugi upit se naziva spoljašnji upit. Pre nego što počne njegovo izvršavanje unutrašnji upit je već završen i u zagradi se nalazi konkretna vrednost rezultata izvršavanja unutrašnjeg upita.

Navedeni mehanizam izvršavanja SELECT naredbe prouzrokuje pojavljivanje i n-torke o Dejanu u rezultujućoj tabeli. U koliko tu n-torku hoćemo da eliminišemo, u WHERE klauzulu spoljašnjeg upita treba dodati uslov: AND IME != 'DEJAN'

Primer 1: Prikazati ime i posao radnika koji rade u Beogradu.

```
SELECT IME, POSLO
FROM RADNIK
WHERE S_RJ = (
    SELECT S_RJ
    FROM R_J
    WHERE GRAD = 'BEOGRAD'
);
```

	IME	POSLO
▶	IVAN	UKOVODILAC
	JOVAN	PREDSEDNIK
	FILIP	SAVETNIK
	DEJAN	ANALITICAR

Slika 1.6 Rezultat SELECT naredbe - primer 1. [Autor]

KORIŠĆENJE FUNKCIJE EXISTS I NOT EXISTS U PODUPITIMA

Rezultat funkcije EXISTS ima vrednost TRUE ako rezultat podupita upita sadrži barem jednu torku za razliku od funkcije NOT EXISTS koja u tom slučaju vraća FALSE

Funkcija EXISTS se u SQL-u koristi za proveru da li je rezultat podupit prazan (ne sadrži torke) ili ne. Rezultat funkcije EXISTS ima Boolean vrednost TRUE ako rezultat podupita sadrži barem jednu torku, ili FALSE ako rezultat podupita ne sadrži nijednu torku.

Primer 2: Naći sve radnike iz tabele RADNIK koji rade u Beogradu?

```
SELECT IME, POSAO
FROM RADNIK
WHERE EXIST(SELECT *
              FROM R_J
              WHERE R_J.S_RJ = RADNIK.S_RJ and GRAD = 'BEOGRAD')
```

Za svaku torku iz RADNIK, se proverava podupit koji selektuje sve torke iz R_J sa istim S_RJ kao u torkama RADNIK koje se nalaze u Beogradu; ako postoji (EXISTS) bar jedna torka u rezultatu podupita, selektuje se ta torka iz RADNIK. Uopšteno, EXISTS vraća TRUE ako postoji barem jedna torka u rezultatu podupita, inače vraća FALSE.

Nasuprot funkciji EXISTS, NOT EXISTS (Q) vraća TRUE ako ne postoje torke u rezultatu podupita, inače vraća FALSE.

Primer 3: Naći sve radnike iz tabele RADNIK koji ne rade u Beogradu?

```
SELECT IME, POSAO
FROM RADNIK
WHERE NOT EXIST(SELECT *
                 FROM R_J
                 WHERE R_J.S_RJ = RADNIK.S_RJ and GRAD = 'BEOGRAD')
```

Za svaku torku iz RADNIK, se proverava podupit koji selektuje sve torke iz R_J sa istim S_RJ kao u torkama RADNIK koje se nalaze u Beogradu; ako ne postoji (NOT EXISTS) nijedna torka u rezultatu podupita, selektuje se ta torka iz RADNIK. Uopšteno, NOT EXISTS vraća TRUE ako ne postoji nijedna torka u rezultatu podupita, inače vraća FALSE.

KORIŠĆENJA PODUPITA ZA SPAJANJE TABELA, PRIMERI 2-3

Primeri pokazuju da spoljašnji i unutrašnji upit mogu biti povezani vrednostima više atributa.

Primer 2: Prikazati ime, posao i lični dohodak radnika u RJ 20 koji imaju isti posao kao radnici RJ projektovanje

```
SELECT IME, POSAO, LD
FROM RADNIK
WHERE S_RJ = 20
AND POSAO IN (
  SELECT POSAO
  FROM RADNIK
  WHERE S_RJ IN (
```

```
SELECT S_RJ
FROM R_J
WHERE NAZIV = 'PROJEKTOVANJE'
)
);
```

Iz ovog upita se dobija sledeći izlaz (Slika 1.7):

	IME	POSAD	LD
►	STEFAN	ANALITICAR	80000
	MARKO	UKOVODILAC	297500
	JELENA	ANALITICAR	110000

Slika 1.7 Rezultat SELECT naredbe - primer 2. [Izvor: Autor]

Sledeći primer pokazuje da spoljašnji i unutrašnji upit mogu biti povezani vrednostima više atributa.

Primer 3: Ko je najbolje plaćeni radnik u svakom odeljenju?

```
SELECT IME, S_RJ, POSAD, LD
FROM RADNIK
WHERE (S_RJ, LD) IN (
    SELECT S_RJ, MAX(LD)
    FROM RADNIK
    GROUP BY S_RJ ASC)
ORDER BY S_RJ;
```

Iz ovog upita se dobija sledeći izlaz (Slika 1.8):

	IME	S_RJ	POSAD	LD
►	JOVAN	10	PREDSEDNİK	500000
	PAVLE	20	SAVETNIK	300000
	ANA	30	UKOVODILAC	285000

Slika 1.8 Rezultat SELECT naredbe - primer 3. [Izvor: Autor]

PRIMER KORIŠĆENJA PODUPITA ZA SPAJANJE TABELA

Korišćenje podupita je povezano sa nekim ograničenjima pa je zato u mnogim situacijama za selektovanje podataka iz više tabela, korisnija upotreba JOIN operacija.

Primer 4: Ko je najbolje plaćeni radnik u svojoj struci, sortirani alfabetski po nazivu struke ?

```
SELECT IME, S_RJ, POSAD, LD
FROM RADNIK
```

```
WHERE (POSAO, LD) IN (
    SELECT POSAO, MAX(LD)
    FROM RADNIK
    GROUP BY POSAO
)
ORDER BY POSAO;
```

Iz ovog upita se dobija sledeći izlaz (*Slika 1.9.*):

	IME	S_RJ	POSAO	LD
▶	DEJAN	10	ANALITICAR	130000
	JOVAN	10	PREDSEDNIK	500000
	MARKO	20	UKOVODILAC	297500
	PAVLE	20	SAVETNIK	300000
	FILIP	10	SAVETNIK	300000
	MILAN	30	TRG PUTNIK	160000

Slika 1.9 Rezultat SELECT naredbe - primer 4. [Izvor: Autor]

Podupiti su veoma moćno sredstvo za spajanje dveju tabela, a li oni imaju i niz **ograničenja**:

1. mogu se selektovati podaci samo iz tabele koja se koristi na najvišem nivou;
2. podupit se ne može koristiti za dobijanje podataka iz više od jedne tabele.

Zbog toga je u mnogim situacijama za selektovanje podataka iz više tabela korisnija upotreba JOIN operacija.

KORIŠĆENJE PODUPITA - VIDEO 1

Video tutorijal za upotrebu podupita

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

UPOTREBA PODUPITA - VIDEO 2

Video tutorijal za korišćenje podupita

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ Poglavlje 2

JOIN za spajanje tabela

JOIN OPERACIJE ZA SPAJANJE VIŠE TABELA

Operacija JOIN, se koristi za kombinovanje povezanih torki iz dve relacije u pojedinačne "duže" torke. Omogućava da obradimo uspostavljene relacije koje postoje među tabelama

Operacija JOIN, se koristi za kombinovanje povezanih torki iz dve relacije u pojedinačne "duže" torke. Ova operacija je veoma važna za bilo koju relacionu bazu podataka sa više od jedne relacije, jer nam omogućava da obradimo uspostavljene relacije (eng. **relationships**) koje postoje među relacijama (eng. **relation**).

Da bi ilustrovali JOIN, pretpostavimo da želimo da pronađemo ime menadžera (iz relacije EMPLOYEE) svakog odeljenja (iz relacije DEPARTMENT). Da bi dobili ime menadžera, potrebno je da kombinujemo svaku toroku odeljenja sa torkom zaposlenog čija Ssn vrednost odgovara Mgr_ssn vrednosti u torci zaposlenog. To radimo koristeći operaciju JOIN i zatim projektujemo rezultat na neophodne attribute. Imajte na umu da je Mgr_ssn strani ključ relacije DEPARTMENT koji se referencira na Ssn, primarni ključ relacije EMPLOYEE. Ovo ograničenje referencijalnog integriteta igra ulogu u upoređivanju torki sa referentnom relacijom EMPLOYEE.

Operacija JOIN može biti specificirana kao operacija CARTESIAN PRODUCT koju prati SELECT operacija. Međutim, JOIN je vrlo važna operacija jer se vrlo često koristi prilikom specificiranja upita za bazu podataka.

Opšti oblik JOIN operacije za dve relacije R (A₁, A₂, ..., A_n) i S (B₁, B₂, ..., B_m) je:

R JOIN[join condition] S

Rezultat JOIN-a je relacija Q sa n + m atributa Q(A₁, A₂, ..., A_n, B₁, B₂, ..., B_m) u datom redosledu; Q ima jednu toroku za svaku kombinaciju torki - jednu iz R i jednu iz S - kad god kombinacija zadovoljava uslov spajanja.

A_iθB_j, A_i je atribut iz R, B_j je atribut iz S, θ je jedan od operatora poređenja { , , ≤, >, ≥, ≠ }.

Torke čiji su atributi po kojima se vrši spajanje NULL ili za koje je uslov spajanja FALSE se ne pojavljuju u rezultatu. U tom smislu, operacija JOIN ne mora nužno čuvati sve informacije u relacijama koje učestvuju u spajanju, jer se torke koje se ne podudaraju sa torkama u drugoj relaciji ne pojavljuju u rezultatu.

VARIJACIJE JOIN: EQUIJOIN I NATURAL JOIN

Kod EQUIJOIN-a se kao jedini operater poređenja koristi =; Kod NATURAL JOIN - dva atributa po kojima se vrši spajanje imaju isto ime u obe relacije;

Najčešća upotreba JOIN-a uključuje samo uslove spajanja koji podrazumevaju poređenje na jednakost. Takav JOIN, gde je jedini operator poređenja koji se koristi =, naziva se **EQUIJOIN**. **Prisetimo da u rezultatu EQUIJOIN uvek imamo jedan ili više para atributa koji imaju identične vrednosti u svakoj torci.**

Na primer, u prethodnom primeru, vrednosti atributa Mgr_ssn i Ssn iz EMPLOYEE i DEPARTMENT su identične u svakoj torci DEPT_MGR (rezultat EQUIJOIN) jer uslov spajanja na jednakost naveden za ova dva atributa zahteva da vrednosti budu identične u svakoj torci koja se pojavljuje u rezultatu. Pošto je jedan od para atributa sa istim vrednostima suvišan, kreirana je nova operacija nazvana **NATURAL JOIN** - označena sa * - kako bi se oslobodili drugog (suvišnog) atributa u EQUIJOIN uslovu. Standardna definicija **NATURAL JOIN** zahteva da dva atributa po kojima se vrši spajanje (ili svaki par atributa po kojima se vrši spajanje) **imaju isto ime u obe relacije**. Ako to nije slučaj, prvo se preimenjuje **operacija preimenovanja**.

Atribut s istim imenom u obe relacije se naziva **atribut spajanja operacije NATURAL JOIN**.

Uopšteno, uslov spajanja za NATURAL JOIN je konstruisan tako što izjednačava svaki par atributa spajanja koji imaju isto ime iz dve relacije i kombinuje ove uslove sa AND. Može postojati lista atributa spajanja iz svake relacije, a svaki odgovarajući par mora imati isto ime.

Generalnija, ali nestandardna definicija za NATURAL JOIN je:

Q NATURAL JOIN R *([list1]),([list2])S

U ovom slučaju, [list1] specificira listu i atributa iz R, a [list2] specificira listu i atributa iz S. Liste se koriste za formiranje uslova poređenja na jednakost između parova odgovarajućih atributa, a uslovi su zatim AND-ovani zajedno. Samo lista koja odgovara atributima prve relacije R - <list1> - se čuva u rezultatu Q.

OSTALE VARIJACIJE JOIN-A

JOIN može biti definisan nad samo jednom relacijom (SELF JOIN) ili N relacija (N-arni JOIN).

Obratite pažnju na to da ako nijedna kombinacija torki ne zadovoljava uslov spajanja, rezultat JOIN-a je prazna relacija sa nula torki. Uopšteno, ako R ima n_R torki i S ima n_S torki, rezultat JOIN operacije

$R \langle \text{join condition} \rangle S$

će imati između **nula i $n_R * n_S$ torki**.

Očekivana veličina rezultata spajanja podeljena sa maksimalnom veličinom $nR * nS$ dovodi do odnosa koji se zove selektivnost spajanja, koji je svojstvo svakog uslova spajanja.

Ako nema uslova spajanja, kvalifikuju se sve kombinacije torki i JOIN prerasta u CARTESIAN PRODUCT, koji se naziva i CROSS PRODUCT ili CROSS JOIN.

Dakle, jedna operacija JOIN se koristi za kombinovanje podataka iz dve relacije, tako da se povezane informacije mogu predstaviti u jednoj tabeli. Ova operacija je takođe poznata kao unutrašnje spajanje (INNER JOIN), kako bi se razlikovala od različitih varijacija spajanja koje se nazivaju spoljašnje spajanje (OUTER JOIN).

Neformalno, unutrašnje spajanje je vrsta operacije poređenja i kombinovanja definisana formalno kao kombinacija CARTESIAN PRODUCT i SELECTION. Imajte na umu da se ponekad spajanje može specificirati nad samo jednom relacijom.

Operacija NATURAL JOIN ili EQUIJOIN takođe može biti specificirana između više tabela, što dovodi do N-arnog spajanja.

PRAVLJENJE UPITA KORIŠĆENJEM JOIN-A

Struktura tabelele RJ

Ovde će biti reči o pravljenju upita korišćenjem JOIN-a koji se može pojaviti kao

1. (INNER) JOIN,
2. CROSS JOIN (CARTESIAN JOIN),
3. LEFT (OUTER) JOIN (LEFTOUTERJOIN),
4. RIGHT (OUTER) JOIN (RIGHT OUTER JOIN),
5. FULL (OUTER) JOIN.
6. SELF JOIN

To će biti ilustrovano primerima upita na primeru relacionog modela sastavljenog od dve relacije:

1. R_J (RADNA_JEDINICA) i
2. RADNIK

Odgovarajući prikaz tabele i podataka u tabeli, kao i unos podataka u SQL-u za datu tabelu:

Relacija (Tabela) R_J (SQL kod):

```
CREATE TABLE `it350l06`.`R_J` (  
  `S_RJ` INT NOT NULL ,  
  `NAZIV` VARCHAR(45) NULL ,  
  `GRAD` VARCHAR(45) NULL ,  
  PRIMARY KEY (`S_RJ`) )  
ENGINE = InnoDB  
DEFAULT CHARACTER SET = utf8;
```

	S_RJ	NAZIV	GRAD
	10	PRODAJA	BEOGRAD
	20	PROIZVODNJA	NIS
	30	PROJEKTOVA...	BOR
	40	ISTRAZIVANJE	NOVI SAD

Slika 2.1 Tabela R_J [Izvor: Autor]

```
INSERT INTO `it350l06`.`R_J` (`S_RJ`, `NAZIV`, `GRAD`)
VALUES (10, 'PRODAJA', 'BEOGRAD');
INSERT INTO `it350l06`.`R_J` (`S_RJ`, `NAZIV`, `GRAD`)
VALUES (20, 'PROIZVODNJA', 'NIS');
INSERT INTO `it350l06`.`R_J` (`S_RJ`, `NAZIV`, `GRAD`)
VALUES (30, 'PROJEKTOVANJE', 'BOR');
INSERT INTO `it350l06`.`R_J` (`S_RJ`, `NAZIV`, `GRAD`)
VALUES (40, 'ISTRAZIVANJE', 'NOVI SAD');
```

TABELE ZA SPAJANJE KORIŠĆENJEM JOIN

Struktura tabele RADNIK

Tabela RADNIK (SQL kod), odgovarajući prikaz tabele i podataka u tabeli, kao i unos podataka u SQL-u za datu tabelu

```
CREATE TABLE `it350l06`.`RADNIK` (
  `S_RADNIK` INT NOT NULL ,
  `S_RJ` INT NULL ,
  `IME` VARCHAR(45) NULL ,
  `POSAO` VARCHAR(45) NULL ,
  `S_RUKO` INT NULL ,
  `DAT_ZAP` DATE NULL ,
  `LD` INT(13) NULL ,
  `PREMIJA` INT(13) NULL ,
  PRIMARY KEY (`S_RADNIK`) ,
  INDEX `S_RJ` (`S_RJ` ASC) ,
  CONSTRAINT `S_RJ`
    FOREIGN KEY (`S_RJ`)
    REFERENCES `it350l06`.`r_j` (`S_RJ`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
```


	S_RADNIK	S_RJ	IME	POSAD	S_RUKO	DAT_ZAP	LD	PREMIJA
▶	3069	20	STEFAN	ANALITICAR	3602	2000-12-17	80000	NULL
	3199	30	MILAN	TRG PUTNIK	3398	3001-02-20	160000	30000
	3221	30	PETAR	TRG PUTNIK	3398	2001-02-21	125000	50000
	3266	20	MARKO	UKOVODILAC	3539	2001-04-02	297500	NULL
	3354	30	MARIJA	TRG PUTNIK	3398	2001-09-01	125000	140000
	3398	30	ANA	UKOVODILAC	3539	2001-05-01	285000	NULL
	3482	10	IVAN	UKOVODILAC	3539	2001-06-09	245000	NULL
	3483	20	PAVLE	SAVETNIK	3266	2001-11-09	300000	NULL
	3539	10	JOVAN	PREDSEDNIK	NULL	2001-11-17	500000	NULL
	3544	30	GORAN	TRG PUTNIK	3398	2001-09-08	150000	0
	3579	20	JELENA	ANALITICAR	3488	2001-09-23	110000	NULL
	3600	30	JANKO	ANALITICAR	3398	2001-12-03	95000	NULL
	3602	10	FILIP	SAVETNIK	3266	2001-12-03	300000	NULL
	3634	10	DEJAN	ANALITICAR	34882	2002-01-23	130000	NULL

Slika 2.2 tabele RADNIK [Izvor: Autor]

```

INSERT INTO `it350l06`.`RADNIK` (`S_RADNIK`, `S_RJ`, `IME`, `POSAD`, `S_RUKO`,
`DAT_ZAP`, `LD`)
VALUES (3069, 20, 'STEFAN', 'ANALITICAR', 3602, '2000-12-17', 80000);

INSERT INTO `it350l06`.`RADNIK` (`S_RADNIK`, `S_RJ`, `IME`, `POSAD`, `S_RUKO`,
`DAT_ZAP`, `LD`, `PREMIJA`)
VALUES (3199, 30, 'MILAN', 'TRG PUTNIK', 3398, '3001-02-20', 160000, 30000);

INSERT INTO `it350l06`.`RADNIK` (`S_RADNIK`, `S_RJ`, `IME`, `POSAD`, `S_RUKO`,
`DAT_ZAP`, `LD`, `PREMIJA`)
VALUES (3221, 30, 'PETAR', 'TRG PUTNIK', 3398, '2001-02-21', 125000, 50000);

INSERT INTO `it350l06`.`RADNIK` (`S_RADNIK`, `S_RJ`, `IME`, `POSAD`, `S_RUKO`,
`DAT_ZAP`, `LD`)
VALUES (3266, 20, 'MARKO', 'UKOVODILAC', 3539, '2001-04-02', 297500);

INSERT INTO `it350l06`.`RADNIK` (`S_RADNIK`, `S_RJ`, `IME`, `POSAD`, `S_RUKO`,
`DAT_ZAP`, `LD`, `PREMIJA`)
VALUES (3354, 30, 'MARIJA', 'TRG PUTNIK', 3398, '2001-09-01', 125000, 140000);

INSERT INTO `it350l06`.`RADNIK` (`S_RADNIK`, `S_RJ`, `IME`, `POSAD`, `S_RUKO`,
`DAT_ZAP`, `LD`)
VALUES (3398, 30, 'ANA', 'UKOVODILAC', 3539, '2001-05-01', 285000);

INSERT INTO `it350l06`.`RADNIK` (`S_RADNIK`, `S_RJ`, `IME`, `POSAD`, `S_RUKO`,
`DAT_ZAP`, `LD`)
VALUES (3482, 10, 'IVAN', 'UKOVODILAC', 3539, '2001-06-09', 245000);

INSERT INTO `it350l06`.`RADNIK` (`S_RADNIK`, `S_RJ`, `IME`, `POSAD`, `S_RUKO`,
`DAT_ZAP`, `LD`)
VALUES (3483, 20, 'PAVLE', 'SAVETNIK', 3266, '2001-11-09', 300000);

INSERT INTO `it350l06`.`RADNIK` (`S_RADNIK`, `S_RJ`, `IME`, `POSAD`, `DAT_ZAP`,
`LD`)
VALUES (3539, 10, 'JOVAN', 'PREDSEDNIK', '2001-11-17', 500000);

```

```
INSERT INTO `it350l06`.`RADNIK` (`S_RADNIK`, `S_RJ`, `IME`, `POSAO`, `S_RUKO`,
`DAT_ZAP`, `LD`, `PREMIJA`)
VALUES (3544, 30, 'GORAN', 'TRG PUTNIK', 3398, '2001-09-08', 150000, 0);

INSERT INTO `it350l06`.`RADNIK` (`S_RADNIK`, `S_RJ`, `IME`, `POSAO`, `S_RUKO`,
`DAT_ZAP`, `LD`)
VALUES (3579, 20, 'JELENA', 'ANALITICAR', 3488, '2001-09-23', 110000);

INSERT INTO `it350l06`.`RADNIK` (`S_RADNIK`, `S_RJ`, `IME`, `POSAO`, `S_RUKO`,
`DAT_ZAP`, `LD`)
VALUES (3600, 30, 'JANKO', 'ANALITICAR', 3398, '2001-12-03', 95000);

INSERT INTO `it350l06`.`RADNIK` (`S_RADNIK`, `S_RJ`, `IME`, `POSAO`, `S_RUKO`,
`DAT_ZAP`, `LD`)
VALUES (3634, 10, 'DEJAN', 'ANALITICAR', 34882, '2002-01-23', 130000);

INSERT INTO `it350l06`.`RADNIK` (`S_RADNIK`, `S_RJ`, `IME`, `POSAO`, `S_RUKO`,
`DAT_ZAP`, `LD`)
VALUES (3602, 10, 'FILIP', 'SAVETNIK', 3266, '2001-12-03', 300000);
```

(INNER) JOIN

U INNER JOIN torka je uključena u rezultat samo ako se podudara sa torkom u drugoj relaciji prema uslovu spajanja.

JOIN (INNER JOIN) povezuje n-torke različitih tabela korišćenjem zajedničkih atributa, odnosno atributa definisanih nad istim domenima. To je uobičajeni način povezivanja tabela relacione baze podataka.

Karakteristika ovog JOIN-a je da se u WHERE klauzuli javlja uslov spajanja. Pored potpune specifikacije naziva atributa po kojima se vrši spajanje tabela, navodi se i operator spajanja, u ovom slučaju jednakost (=).

Spajanje tabela moguće je vršiti u odnosu na bilo koji operator poređenja. Za spajanje tabela se može koristiti non-ANSI join sintaksa kada se tabele koje se spajaju navode uz klauzulu FROM dok se uslov spajanja definiše kao predikat u klauzuli WHERE

```
SELECT ime, naziv, grad
FROM RADNIK, R_J
WHERE RADNIK.S_RJ = R_J.S_RJ;
```

Osim korišćenja non-ANSI join sintakse, može se koristiti ANSI join sintaksa na sledeći način.

```
SELECT ime, grad
FROM RADNIK INNER JOIN R_J
ON RADNIK.S_RJ = R_J.S_RJ;
```

Tabele koje se spajaju kao i sam JOIN se nalaze u FROM klauzuli tako da se WHERE klauzula sadrži samo filtere a ne i uslove spajanja.

U INNER JOIN torka je uključena u rezultat samo ako se podudara sa torkom u drugoj relaciji prema uslovu spajanja.

Na primeru datog upita, u rezultat su uključeni samo radnici za koje postoji šifra radne jedinice (S_RJ) i ona se podudara sa šifrom radne jedinice (S_RJ) u tabeli R_J; svi redovi iz RADNIK za koje je šifra radne jedinice NULL ili ona ne postoji u radnoj jedinici R_J bi bili isključeni.

Ovaj način spajanja, kod koga je torka je uključena u rezultat samo ako se podudara sa torkom u drugoj relaciji naziva se INNER JOIN, (unutrašnja veza) i može sesmatrati podrazumevanim tipom spajanja.

PRIMERI ZA (INNER) JOIN

Primer JOIN-a nad tabelama ORDER i CUSTOMER

JOIN treba napraviti na osnovu dve tabele ORDER i CUSTOMER i čije su strukture date na slikama 2.3 i 2.4.

OrderID	CustomerID	OrderDate
10308	2	1996-09-18
10309	37	1996-09-19
10310	77	1996-09-20

Slika 2.3 Sadržaj tabele ORDER [Izvor: NM IT350-2020/2021.]

CustomerID	CustomerName	ContactName	Country
1	Alfreds Futterkiste	Maria Anders	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mexico

Slika 2.4 Sadržaj tabele CUSTOMER [Izvor: NM IT350-2020/2021.]

Napomenimo da kolona "CustomerID" u tabeli ORDER predstavlja strani ključ čiji sadržaj mora biti isti kao i sadržaj kolone "CustomerID" u tabeli CUSTOMER. Tada se korišćenjem SQL naredbe koja sadrži JOIN mogu selektovati redovi kod kojih se vrednosti kolone "CustomerID" poklapaju u obe tabele.

```
SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate
FROM Orders INNER JOIN Customers
ON Orders.CustomerID=Customers.CustomerID;
```

Tako će se dobiti sledeći redovi tabele:

OrderID	CustomerName	OrderDate
10308	Ana Trujillo Emparedados y helados	9/18/1996
10365	Antonio Moreno Taquería	11/27/1996
10383	Around the Horn	12/16/1996
10355	Around the Horn	11/15/1996
10278	Berglunds snabbköp	8/12/1996

Slika 2.5 Izlaz iz upita koji sadrži JOIN [Izvor: NM IT350-2020/2021.]

JOIN za povezivanje tri tabele:

```
SELECT Orders.OrderID, Customers.CustomerName, Shippers.ShipperName
FROM ((OrdersINNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID)
INNER JOIN Shippers ON Orders.ShipperID = Shippers.ShipperID)
```

INNER JOIN: PRIMERI SA [HTTPS://DOCS.ORACLE.COM](https://docs.oracle.com)

Primeri 1-4 INNER JOIN sa <https://docs.oracle.com>

Primer 1: Spojite tabele EMP_ACT i EMPLOYEE tako što ćete selektovati sve kolone iz tabele EMP_ACT i njima dodati LASTNAME iz tabele EMPLOYEE

```
SELECT SAMP.EMP_ACT.*, LASTNAME
FROM SAMP.EMP_ACT JOIN SAMP.EMPLOYEE
ON EMP_ACT.EMPNO = EMPLOYEE.EMPNO
```

Primer 2: Spojite tabele EMPLOYEE i DEPARTMENT tako što ćete selektovati EMPNO i LASTNAME, WORKDEPT iz tabele EMPLOYEE i DEPTNO i DEPTNAME iz tabele DEPARTMENT za sve zaposlene koji su rođeni (BIRTHDATE) pre 1930.

```
SELECT EMPNO, LASTNAME, WORKDEPT, DEPTNAME
FROM SAMP.EMPLOYEE JOIN SAMP.DEPARTMENT
ON WORKDEPT = DEPTNO
AND YEAR(BIRTHDATE) < 1930
```

Primer 3: Izlistati sva odeljenja sa brojevima zaposlenih i prezimenom rukovodioca (menadžera)

```
SELECT DEPTNO, DEPTNAME, EMPNO, LASTNAME
FROM DEPARTMENT INNER JOIN EMPLOYEE
ON MGRNO = EMPNO
```

Primer 4: Izlistati brojeve i prezimena svih zaposlenih sa brojem zaposlenog i prezimenom njihovih rukovodioca (menadžera)

```
SELECT E.EMPNO, E.LASTNAME, M.EMPNO, M.LASTNAME
FROM EMPLOYEE E INNER JOIN
DEPARTMENT INNER JOIN EMPLOYEE M
ON MGRNO = M.EMPNO
ON E.WORKDEPT = DEPTNO
```

JOIN VIDEO

Video tutorijal za operaciju JOIN

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

CROSS JOIN

U WHERE klauzuli je izostavljen uslov spajanja

CROSS JOIN se izvršava ukoliko se u WHERE klauzuli izostavi uslov spajanja. Ako se primenjuje na dve tabele, tada se svaka n-torka prve spaja sa svakom n-torkom druge tabele. Ako se primenjuje na više od dve tabele, tada se rezultujuća tabela sa stoji od svih kombinacija n-torki svih tabela.

Korišćenjem CROSS JOIN nad dve tabele se dobija DEKARTOV PROIZVOD tabela pri čemu se spaja svaka torka jedne tabele sa svakom torkom druge tabele.

Primer: Ponovimo prethodni primer izostavljajući JOIN klauzulu

```
SELECT ime, grad
FROM RADNIK CROSS JOIN R_J;
```

Alternativno, možemo da koristimo i

```
SELECT ime, grad
FROM RADNIK, R_J;
```

ili i

```
SELECT ime, grad
FROM RADNIK JOIN R_J;
```

Primer prikazuje da primena Dekartovog proizvoda u ovom slučaju nema smisla. U nekim slučajevima, međutim, izbor tog načina realizacije zahteva je veoma elegantno rešenje. Ukoliko svakoj n-torci tabele sa velikim brojem n-torki treba pridružiti jednu istu vrednost, moguće je formirati pomoćnu tabelu sa jednom kolonom i tom vrednošću u koloni, i izvršiti Dekartov proizvod te dve tabele.

Kao što je rečeno CROSS JOIN daje Dekartov proizvod redova tabela, i pošto nema kolona po kojima se vrši spajanje, dobijaju se sve moguće kombinacije redova dve tabele.

▼ Poglavlje 3

LEFT (OUTER) JOIN

ŠTA JE OUTER JOIN?

U OUTER JOIN-u su uključene sve torke relacije, a ne samo one koje se podudaraju sa torkama u drugoj relaciji prema uslovu spajanja kao što je to slučaj kod INNER JOIN

Ako korisnik zahteva da u rezultat JOIN-a budu uključeni sve torke relacije (leve, desne ili obe relacije) a ne samo one koje se podudaraju sa torkama u drugoj relaciji prema uslovu spajanja kao što je to slučaj kod INNER JOIN, eksplicitno mora biti korišćen OUTER JOIN.

Postoji nekoliko tipova OUTER JOIN-a:

1. LEFT OUTER JOIN: u rezultatu JOIN-a mora da se pojavi svaka torka iz leve relacije, koja može biti ispunjena NULL vrednostima za attribute desne relacije,
2. RIGHT OUTER JOIN u rezultatu mora da se pojavi svaka torka iz desne relacije, ako nema odgovarajuće torke, ispunjena je NULL vrednostima za attribute leve relacije
3. FULL OUTER JOIN. U rezultatu se pojavljuju sve torke iz leve relacije koje mogu biti ispunjene NULL vrednostima za attribute desne relacije kao i sve torke iz desne relacije koje mogu biti ispunjene NULL vrednostima za attribute leve relacije

U zadnje tri opcije, ključna reč OUTER može biti izostavljena.

ŠTA DAJE LEFT (OUTER) JOIN?

Kompletni skup zapisa iz leve tabele sa odgovarajućim podacima iz desne tabele koji se podudarju sa levom tabelom

LEFT (OUTER) JOIN daje kompletni skup zapisa iz tabele A (leve tabele sa odgovarajućim podacima iz tabele B (desne tabele) koji se podudaraju sa levom tabelom A.(slika 1.)

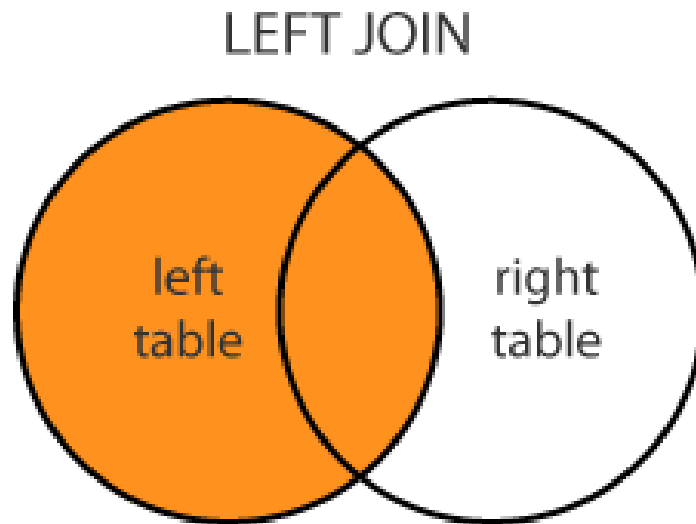
Korišćenje reči OUTER je opciono.

Ukoliko nema podudaranja, desna strana će sadržati NULL vrednosti

```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name=table2.column_name;
```

Takođe, možemo dodati ključulu OUTER

```
SELECT column_name(s)
FROM table1
LEFT OUTER JOIN table2
ON table1.column_name=table2.column_name;
```



Slika 3.1 . Prikaz Left JOIN [Izvor: NM IT250-2020/2021.]

Ako korisnik zahteva da budu uključeni sve torke relacije u JOIN-u a ne samo one koje se podudaraju sa torkama u drugoj relaciji prema uslovu spajanja kao što je to slučaj kod INNER JOIN, eksplicitno mora biti korišćen OUTER JOIN. **LEFT OUTER JOIN: u rezultatu mora da se pojavi svaka torka iz leve relacije, koja može biti ispunjena NULL vrednostima za atribute desne relacije.**

LEFT JOIN - PRIMERI

Primeri LEFT JOIN-a nad tabelama R_J i RADNIK odnosno ORDER i CUSTOMER

Primer 1: Naredba:

```
SELECT R_J. NAZIV, RADNIK.IME
FROM R_J LEFT JOIN RADNIK
ON R_J.S_RJ = RADNIK.S_RJ
```

daće sve redove iz leve tabele (R_J) čak i ako za njih nema odgovarajućih redova u desnoj tabeli RADNIK (pojaviće se redovi sa nazivima radnih jedinica koje nemaju radnike)

Non-ANSI ekvivalent equivalent prethodnoj naredbi je:

```
SELECT R_J. NAZIV, RADNIK.IME
```

```
FROM R_J, RADNIK
WHERE R_J.S_RJ = RADNIK.S_RJ (+)
```

Napomenoi da se "(+)" koristi kako bi se označila strana join uslova koji može nedostajati. Ukoliko se spajanje vrši po više kolona svaka kolona mora da ima "(+)" . Za razliku od ANSI join sintakse, u non-ANSI join sintaksi nije bitan redosled tabela.

Primer 2: Ako koristimo tabele CUSTOMER i ORDER na slikama 3.2 i 3.3 tada će naredba:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico

Slika 3.2 Tabela CUSTOMER [Izvor: NM IT350-2020/2021.]

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10308	2	7	1996-09-18	3
10309	37	3	1996-09-19	1
10310	77	8	1996-09-20	2

Slika 3.3 Tabela ORDER [Izvor: NM IT350-2020/2021.]

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
ORDER BY Customers.CustomerName;
```

vratiti sve redove iz leve tabele CUSTOMER bez obzira da li za kupce postoje narudžbine u desnoj tabeli ORDER.

LEFT OUTER JOIN - PRIMERI SA [HTTPS://DOCS.ORACLE.COM](https://docs.oracle.com)

Primeri 1-3 LEFT OUTER JOIN sa <https://docs.oracle.com>

Primer 1: Upariti gradove i zemlje u Aziji

```
SELECT CITIES.COUNTRY, CITIES.CITY_NAME, REGION
FROM Countries
LEFT OUTER JOIN Cities
ON CITIES.COUNTRY_ISO_CODE = COUNTRIES.COUNTRY_ISO_CODE
WHERE REGION = 'Asia'
```


Primer 2: Spojiti tabele *EMPLOYEE* i *DEPARTMENT*, tako što ćete selektovati *EMPNO*, *LASTNAME*, *WORKDEPT* iz tabele *EMPLOYEE* i *DEPTNO* i *DEPTNAME* iz tabele *DEPARTMENT* za sve zaposlene koji su rođeni (*BIRTHDATE*) pre 1930

```
SELECT EMPNO, LASTNAME, WORKDEPT, DEPTNAME
FROM SAMP.EMPLOYEE LEFT OUTER JOIN SAMP.DEPARTMENT
ON WORKDEPT = DEPTNO
AND YEAR(BIRTHDATE) < 1930
```

Primer 3: Izlistati sva odeljenja sa employee number i last name rukovodioca, uključujući i odeljenja koja nemaju menadžere

```
SELECT DEPTNO, DEPTNAME, EMPNO, LASTNAME
FROM DEPARTMENT LEFT OUTER JOIN EMPLOYEE
ON MGRNO = EMPNO
```

▼ Poglavlje 4

RIGHT (OUTER) JOIN

ŠTA DAJE RIGHT (OUTER) JOIN?

Kompletan skup zapisa iz desne tabele čak i ako za njih nema odgovarajućih redova u levoj tabeli

Naredba za prikaz zapisa iz desne tabele bez obzira na to što za njih nema odgovarajućih redova u levoj tabeli.

```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name=table2.column_name;
```

Drugi način za prikaz kao u prethodnom primeru.

```
SELECT column_name(s)
FROM table1
RIGHT OUTER JOIN table2
ON table1.column_name=table2.column_name;
```

RIGHT JOIN - PRIMERI

Primer 1: nad tabelama R_J i RADNIK; Primer 2: nad tabelama ORDER i EMPLOYEEER

Primer 1: Naredba:

```
SELECT R_J. NAZIV, RADNIK.IME
FROM RADNIK
RIGHT R_J ON R_J.S_RJ = RADNIK.S_RJ
```

daće sve redove iz desne tabele (RADNIK) čak i ako za njih nema odgovarajućih redova u levoj tabeli R_J (pojaviće se redovi sa imenima radnika za koje ne postoje radne jedinice)

Napomenimo da non-ANSI outer join sintaksa ne zavisi od redosleda tabela tako da se ne može razlikovati LEFT i RIGHT join već postoji samo OUTER join.

Primer 2: SELECT naredba nad tabelama EMPLOYEE i ORDER na slikama 4.1 i 4.2, sa klauzulom RIGHT JOIN

```
SELECT Orders.OrderID, Employees.LastName, Employees.FirstName
FROM Orders
RIGHT JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID
ORDER BY Orders.OrderID;
```

EmployeeID	LastName	FirstName	BirthDate	Photo
1	Davolio	Nancy	12/8/1968	EmpID1.pic
2	Fuller	Andrew	2/19/1952	EmpID2.pic
3	Leverling	Janet	8/30/1963	EmpID3.pic

Slika 4.1 Tabela EMPLOYEEER [Izvor: NM IT350-2020/2021.]

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10308	2	7	1996-09-18	3
10309	37	3	1996-09-19	1
10310	77	8	1996-09-20	2

Slika 4.2 Tabela ORDER [Izvor: NM IT350-2020/2021.]

vraća sve redove desne tabele RADNIK bez obzira da li se oni podudaraju sa levom tabelom ORDER (imena svih radnika bez obzira da su plasirali porudžbenicu ili ne)

RIGHT OUTER JOIN - PRIMERI SA [HTTPS://DOCS.ORACLE.COM](https://docs.oracle.com)

Primeri 1-3 RIGHT OUTER JOIN sa <https://docs.oracle.com>

Primer 1: Nađi sve zemlje i odgovarajuće gradove uključujući i zemlje bez i jednog grada

```
SELECT COUNTRIES.COUNTRY, CITIES.CITY_NAME
FROM CITIES
RIGHT OUTER JOIN COUNTRIES
ON CITIES.COUNTRY_ISO_CODE = COUNTRIES.COUNTRY_ISO_CODE
```

Primer 2: Nađi sve zemlje u Africi i odgovarajuće gradove uključujući i zemlje bez i jednog grada

```
SELECT COUNTRIES.COUNTRY, CITIES.CITY_NAME
FROM CITIES
RIGHT OUTER JOIN COUNTRIES
ON CITIES.COUNTRY_ISO_CODE = COUNTRIES.COUNTRY_ISO_CODE
WHERE Countries.region = 'Africa'
```

Spojene tabele se mogu koristiti u drugoj JOIN operaciji pa se u FROM klauzuli može koristiti multiple join operacija

Primer 3: Izlistati employee number i last name sa employee number i last name njihovih rukovodioca (menager)

```
SELECT E.EMPNO, E.LASTNAME, M.EMPNO, M.LASTNAME  
FROM EMPLOYEE E RIGHT OUTER JOIN  
DEPARTMENT RIGHT OUTER JOIN EMPLOYEE M  
ON MGRNO = M.EMPNO  
ON E.WORKDEPT = DEPTNO
```

▼ Poglavlje 5

FULL (OUTER) JOIN

ŠTA DAJE FULL (OUTER) JOIN?

Daje sve redove i iz leve i iz desne tabele

FULL (OUTER) JOIN selektuje sve redove koji se nalaze i u levoj i u desnoj tabeli.

FULL (OUTER) JOIN kombinuje redove i iz LEFT i iz RIGHT join-a.

Korišćenje reči OUTER je opciono.

U ovom slučaju ne postoji uobičajeno uparivanje. Ukoliko bilo koja strana, leva ili desna nema podatke za uparivanje, pojaviće se NULL vrednosti a redovi neće izostati.

```
SELECT column_name(s)
FROM table1
FULL OUTER JOIN table2
ON table1.column_name=table2.column_name;
```

FULL JOIN - PRIMERI

Primer 1: nad tabelama R_J i RADNIK; Primer 2: nad tabelama ORDER i EMPLOYEEER

Primer 1: Naredba:

```
SELECT R_J.NAZIV, RADNIK.IME
FROM RADNIK
FULL OUTER JOIN S_RJ ON R_J.S_RJ = RADNIK.S_RJ
```

daće imena svih radnika sa odgovarajućim imenima radnih jedinica u kojima rade, imena svih radnika koje nemaju unete radne jedinice kao i imena svih radnih jedinica za u kojima ne postoje radnici.

Primer 2: Naredbom SELECT koja sadrži klauzulu FULL JOIN nad tabelama CUSTOMER i ORDER na slikama 2. i 3.

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico

Slika 5.1 Tabela CUSTOMER [Izvor: NM IT350-2020/2021.]

OrderID	CustomerID	OrderDate
10308	2	1996-09-18
10309	37	1996-09-19
10310	77	1996-09-20

Slika 5.2 Tabela ORDER [Izvor: NM IT350-2020/2021.]

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
FULL OUTER JOIN Orders ON Customers.CustomerID=Orders.CustomerID
ORDER BY Customers.CustomerName;
```

će se dobiti sledeći izlaz:

CustomerName	OrderID
Alfreds Futterkiste	
Ana Trujillo Emparedados y helados	10308
Antonio Moreno Taquería	10365
	10382
	10351

Slika 5.3 Izlaz koji se dobija korišćenjem FULL JOIN [Izvor: NM IT350-2020/2021.]

▼ Poglavlje 6

SELF JOIN

KADA SE KORISTI SELF JOIN?

SELF JOIN spaja redove tabele sa ostalim redovima te iste tabele

SELF JOIN omogućava spajanje tabele sa samom sobom po kolonama koje sadrže isti tip informacija. SELF JOIN spaja redove tabele sa ostalim redovima te iste tabele.

Primer 1: Prikaži ime i posao svakog radnika koji ima pretpostavljenog. Takođe prikaži ime i posao pretpostavljenog.

```
SELECT PODR.IME, PODR.POSAO, PODR.S_RUKO,  
NADR.S_RADNIK SEF, NADR.IME SEF_IME,  
NADR.POSAO SEF_POSAO  
FROM RADNIK PODR, RADNIK NADR  
WHERE PODR.S_RUKO = NADR.S_RADNIK;
```

	IME	POS AO	S_RUKO	SEF	SEF_IME	SEF_POSAO
▶	STEFAN	ANALITICAR	3602	3602	FILIP	SAVETNIK
	MILAN	TRG PUTNIK	3398	3398	ANA	UKOVODILAC
	PETAR	TRG PUTNIK	3398	3398	ANA	UKOVODILAC
	MARKO	UKOVODILAC	3539	3539	JOVAN	PREDSEDNİK
	MARIJA	TRG PUTNIK	3398	3398	ANA	UKOVODILAC
	ANA	UKOVODILAC	3539	3539	JOVAN	PREDSEDNİK
	IVAN	UKOVODILAC	3539	3539	JOVAN	PREDSEDNİK
	PAVLE	SAVETNIK	3266	3266	MARKO	UKOVODILAC
	GORAN	TRG PUTNIK	3398	3398	ANA	UKOVODILAC
	JANKO	ANALITICAR	3398	3398	ANA	UKOVODILAC
	FILIP	SAVETNIK	3266	3266	MARKO	UKOVODILAC
	DEJAN	ANALITICAR	3482	3482	IVAN	UKOVODILAC

Slika 6.1 Rezultat za primer 1. [Izvor: Autor]

Da bi se realizovao ovaj upit, tabeli RA DNIK se daju dva sinonima

– **PODR** i **NADR**. Kada su potrebni podaci o radnicima, tabela RA DNIK se referencira sinonimom PODR, a kada su potrebni podaci o neposrednim rukovodiocima – sinonimom NADR. Na taj način, pomoću navedenih sinonima, praktično su formirane dve virtuelne tabele sa identičnim sadržajem.

JOIN - OUTER JOIN VIDEO 2

Video tutorijal 2 za JOIN-OUTER JOIN

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

✓ Poglavlje 7

Korišćenje pogleda (eng. VIEW)

TABELE ZA KREIRANJE VIEW-A

CUSTOMER, ARTIST i CUSTOMER_ARTIST_INT

Rad sa pogledima je moguć korišćenjem naredbi

1. CREATE VIEW,
2. UPDATE VIEW i
3. DROP VIEW.

U SQL-u, pogled (VIEW) je virtualna tabela koja se dobija kao rezultat skupa SQL naredbi. Za kreiranje VIEW-a se koristi naredba CREATE VIEW

Pogled kao i tabela sadrži redove i kolone. Kolone u pogledu su polja iz jedne ili više realnih tabela u bazi podataka.

Za kreiranje VIEW-a u će biti korišćene table CUSTOMER, ARTIST i CUSTOMER_ARTIST_INT, radi lakšeg razumevanja na slici 7.1 je prikazan sadržaj tih tabela

CustomerID	Name	Street	City	State	ZipPostalCode
1000	Jeffrey Janes	123 W. Elm St	Renton	WA	98123
1001	David Smith	813 Tumbleweed L	Loveland	CO	80345
1015	Tiffany Twilight	88 - First Avenue	Langley	WA	98114
1033	Fred Smathers	10899 - 88th Ave	Bainbridge Island	WA	98108
1034	Mary Beth Frederik	25 South Lafayette	Denver	CO	80210
1036	Selma Warning	205 Burnaby	Vancouver	BC	V0N 1B4

Country	AreaCode	PhoneNumber	Email
USA	206	555-1345	Customer1000@somewhere.com
USA	303	555-5434	Customer1001@somewhere.com
USA	206	555-1000	Customer1015@somewhere.com
USA	206	555-1234	Customer1033@somewhere.com
USA	303	555-1000	Customer1034@somewhere.com

Tabela: CUSTOMER

ArtistID	Name	Nationality	Birthdate	DeceasedDate
3	Miro	Spanish	1870	1950
4	Kandinsky	Russian	1854	1900
5	Frings	US	1950	<NULL>
6	Klee	German	1900	<NULL>
8	Moos	US	<NULL>	<NULL>

Tabela: ARTIST

ArtistID	CustomerID
3	1036
5	1015
5	1034
5	1041
5	1051
8	1034

Slika 7.1 Tabele za kreiranje pogleda CUSTOMER i ARTIST [Izvor: NM IT350-2020/2021.]

NAREDBE ZA KREIRANJE TABELA CUSTOMER, ARTIST

CUSTOMER, ARTIST

```
CREATE TABLE customer(  
    CustomerID INT NOT NULL,  
    Name VARCHAR(45) NOT NULL,  
    Street VARCHAR(45) NULL,  
    City VARCHAR(45) NULL,  
    State VARCHAR(45) NULL,  
    ZipPostalCode VARCHAR(45) NULL,  
    Country VARCHAR(45) NULL,  
    AreaCode VARCHAR(45) NULL,  
    PhoneNumber VARCHAR(45) NULL,  
    Email VARCHAR(45) NULL,  
  
    CONSTRAINT CustomerPK  
    PRIMARY KEY(CustomerID),  
    CONSTRAINT EmailAK1  
    UNIQUE(Email)  
);
```

```
CREATE TABLE artist(  
    ArtistID INT NOT NULL,  
    Name VARCHAR(45) NOT NULL,  
    Nationality VARCHAR(45) NULL,  
    Birthdate NUMERIC(4,0) NULL,  
    DeceasedDate NUMERIC(4,0) NULL,  
  
    CONSTRAINT ArtistPK  
    PRIMARY KEY(ArtistID),  
    CONSTRAINT ArtistAK1 UNIQUE(Name),  
    CONSTRAINT NationlitiValues  
    CHECK (Nationality  
    IN ('Canadian', 'English',  
    'French', 'German',  
    'Russian', 'Mexican',  
    'Spanish', 'US')),  
    CONSTRAINT BirthValuesCheck  
    CHECK (Birthdate < DeceasedDate),  
    CONSTRAINT ValidBirthYear CHECK  
    (Birthdate  
    LIKE '[1-2][0-9][0-9][0-9]'),  
    CONSTRAINT ValidDeathYear CHECK  
    (DeceasedDate  
    LIKE '[1-2][0-9][0-9][0-9]')  
);
```

KREIRANJE TABELA CUSTOMER_ARTIST_INT, WORK

CUSTOMER_ARTIST_INT, WORK

```
CREATE TABLE Customer_Artist_Int(  
    caID INT NOT NULL AUTO_INCREMENT,  
    ArtistID INT NOT NULL,  
    CustomerID INT NOT NULL,  
  
    CONSTRAINT CustArtistPK  
    PRIMARY KEY (caID),  
    CONSTRAINT CustArtIntArtistFK  
    FOREIGN KEY (ArtistID)  
    REFERENCES artist(ArtistID)  
    ON UPDATE NO ACTION ON DELETE CASCADE,  
    CONSTRAINT CustArtIntCustomerFK  
    FOREIGN KEY (CustomerID)  
    REFERENCES customer(CustomerID)  
    ON UPDATE NO ACTION ON DELETE CASCADE  
);
```

```
CREATE TABLE work(  
    WorkID INT NOT NULL,  
    Title VARCHAR(145) NOT NULL,  
    Copy VARCHAR(145) NOT NULL,  
    Description VARCHAR(1255) NULL  
    DEFAULT 'Unknown provenance',  
    ArtistID INT NOT NULL,  
  
    CONSTRAINT WorkPK  
    PRIMARY KEY(WorkID),  
    CONSTRAINT WorkAK  
    UNIQUE(Title, Copy),  
    CONSTRAINT ArtistFK  
    FOREIGN KEY(ArtistID)  
    REFERENCES artist(ArtistID)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION  
);
```

NAREDBE ZA KREIRANJE TABELA TRANSACTIONS

TRANSACTIONS

```
CREATE TABLE transactions(  
    TransactionID INT NOT NULL,  
    DataAquired DATE NOT NULL,  
    AquisitionPrice NUMERIC(8,2) NOT NULL,
```

```
PurchaseDate DATE NULL,
SalesPrice NUMERIC(8,2) NULL,
AskingPrice NUMERIC(8,2) NULL,
CustomerID INT NULL,
WorkID INT NOT NULL,

CONSTRAINT TransactionPK
PRIMARY KEY(TransactionID) ,
CONSTRAINT ValidPD
CHECK (PurchaseDate > DataAquired),
CONSTRAINT SalesPR
CHECK ((SalesPrice > 1000)
AND (SalesPrice <= 200000)).
CONSTRAINT TransWorkFK
FOREIGN KEY (WorkID)
REFERENCES work(WorkID)
ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT TransCustFK
FOREIGN KEY (CustomerID)
REFERENCES customer(CustomerID)
ON UPDATE NO ACTION ON DELETE NO ACTION;
);
```

VIEW ZA SAKRIVANJE NEKE KOLONE ILI REDOVA TABELE

VIEW se može koristiti da se neke kolone sakriju kako bi se uprostila tabela ili sprečilo prikazivanje osetljivih podataka

VIEW se može koristiti da se neke kolone sakriju kako bi se uprostila tabela ili sprečilo prikazivanje osetljivih podataka.

Primer 1: Pretpostavimo da želimo da uprostimo listu kupaca galerije slika pr iskazivanjem samo njihovog imena i broja telefona.

```
CREATE VIEW BasicCustomerData AS
SELECT Name, AreaCode, PhoneNumber
FROM CUSTOMER;
```

Rezultat **SELECT** -a iz ovog **VIEW** -a je prikazan na slici 7.2 :

Name	AreaCode	PhoneNumber
Jeffrey Janes	206	555-1345
David Smith	303	555-5434
Tiffany Twilight	206	555-1000
Fred Smathers	206	555-1234
Mary Beth Frederickson	303	555-1000

Slika 7.2 Rezultat za VIEW - primer 1 [Izvor: NM IT350-2020/2021.]

Ako menadžer galerije želi da sakrije kolone AcquisitionPrice i SalesPrice u TRANSACTION, on može kreirati view koji neće uključiti te dve kolone. **VIEW** -om se takođe mogu klauzulom **WHERE** sakriti redovi tabela

Primer 2: Kreirati view sa imenima i brojevima telefona kupaca sa adresama u Washingtonu.

```
CREATE VIEW BasicCustomerData_WA AS  
SELECT Name, AreaCode, PhoneNumber  
FROM CUSTOMER  
WHERE State = 'WA';
```

Na slici 7.3 je prikazan rezultat VIEW-a.

Name	AreaCode	PhoneNumber
Jeffrey Janes	206	555-1345
Tiffany Twilight	206	555-1000
Fred Smathers	206	555-1234
Chris Wilkens	206	555-1234

Slika 7.3 Rezultat za VIEW - primer 2 [Izvor: NM IT350-2020/2021.]

VIEW ZA PRIKAZ REZULTATA NEKIH IZRAČUNAVANJA

Svrha view-ova je da se prikažu rezultati izračunatih kolona bez obaveze korisnika da unese izraz za izračunavanje

Svrha VIEW-ova je da se prikažu rezultati izračunatih kolona bez obaveze korisnika da unese izraz za izračunavanje.

Primer 3: Kreirati view u kojem je kombinovan AreaCode i PhoneNumber. Izlaz iz VIEW-a

```
CREATE VIEW CustomerPhone AS  
SELECT Name, '(' || AreaCode || ')' PhoneNumber Phone  
FROM CUSTOMER;
```

Naredbom:

```
SELECT * FROM CustomerPhone;
```

se dobija rezultat prikazan na narednoj slici 7.4 :

Name	Phone
Jeffrey Janes	(206) 555-1345
David Smith	(303) 555-5434
Tiffany Twilight	(206) 555-1000
Fred Smathers	(206) 555-1234
Mary Beth Frederickson	(303) 555-1000
Selma Warning	(253) 555-1234
Susan Wu	(721) 555-1234

Slika 7.4 Rezultat za VIEW - primer 3 [Izvor: NM IT350-2020/2021.]

VIEW ZA SLOJEVITO UGRAĐIVANJE NEKE FUNKCIJE

Koristi se kada u WHERE klauzuli naredbe SELECT treba koristiti funkcije ili izraze kojima se nešto sračunava

Kao što je rečeno, u WHERE klauzuli SQL naredbe se ne mogu koristiti funkcije ili izrazi kojima se nešto izračunava. Umesto toga, treba kreirati VIEW u kojem je to izračunavanje obavljeno, a zatim SELECT naredbu koja u WHERE klauzuli koristi taj VIEW.

Primer 4:

```
CREATE VIEW ArtistWorkNet AS
SELECT W.WorkID, Name, Title, Copy,
AcquisitionPrice, SalesPrice,
(SalesPrice - AcquisitionPrice) NetPrice
FROM TRANSACTION T, WORK W, ARTIST A
WHERE T.WorkID = W.WorkID
AND A.ArtistID = W.ArtistID;
```

Ovim VIEW-om se pravi JOIN između TRANSACTION, WORK i ARTIST i sračunava kolona NetPrice koja se može koristiti u WHERE klauzuli

```
SELECT Name, NetPrice
FROM ArtistWorkNet
WHERE NetPrice > 10000;
```

Ova slojevitost se može primeniti na više nivoa.

Primer 5: Možemo kreirati drugi VIEW u kojem će se izvršiti sračunavanje nad sračunatim vrednostima iz prvog view-a.

```
CREATE VIEW WorkNet AS
SELECT Name, Title, Copy,
SUM(NetPrice) TotalNet
FROM ArtistWorkNet
GROUP BY Name, Title, Copy;
```

Izračunata kolona TotalNet se može sada koristiti u WHERE klauzuli SELECT naredbe nad WorkNet:

```
SELECT * FROM WorkNet  
WHERE TotalNet < 25000;
```

čime se dobija izlaz prikazan na slici 7.5:

Name	NetPrice
Tobey	17800.00
Miro	34800.00
Tobey	24500.00

Slika 7.5 Rezultat za VIEW - primer 5 [Izvor: NM IT350-2020/2021.]

VIEW ZA IZOLACIJU IZVORNIH TABELA OD APLIKATIVNOG KODA

VIEW-ovi se mogu koristiti za izolaciju izvornih tabela od aplikativnog koda

VIEW-ovi se mogu koristiti za izolaciju izvornih tabela od aplikativnog koda.

Primer 6:

```
CREATE VIEW CustomerTable1 AS  
SELECT * FROM CUSTOMER;
```

VIEW-om smo tabeli CUSTOMER dodelili alijas CustomerTable1.

Ako aplikativni kod u SQL-u koristi CustomerTable1, tada je stvarni izvor podataka sakriven od aplikativnih programera. Takva izolacija tabela omogućava fleksibilnost administratoru baze podataka.

Primer 7: Ako se u budućnosti izvor podataka o kupcima promeni, pa se podaci ne uzimaju iz tabele CUSTOMER već iz druge tabele pod nazivom NEW_CUSTOMER, administrator baze podataka treba da samo redefiniše view CustomerTable1 na sledeći način:

```
CREATE VIEW CustomerTable1 AS  
SELECT * FROM NEW_CUSTOMER;
```

Sve će se aplikacije izvršavati nad tabelom CustomerTable1 koja sada ima drugi izvor podataka.

VIEW-ovi se mogu koristiti i za dodeljivanje različitih skupova dozvola za obradu iste tabele.

Primer 8: za tabelu CUSTOMER se može kreirati:

1. CustomerRead VIEW koji ima samo dozvolu čitanja,

2. CustomerUpdate sa dozvolom čitanja i ažuriranja itd.

Aplikacija koja ne treba da ažurira podatke, može da koristi CustomerRead dok one koje imaju potrebu da ih ažuriraju view CustomerUpdate.

VIEW-ovi se takođe mogu koristiti za definisanje različitih skupova trigera nad istim izvorom podataka. Ova tehnika se naročito koristi za realizaciju O – M i M – M veza.

U tom slučaju se kreira jedan VIEW koji ima skup trigera kojim se zabranjuje brisanje dece i drugi VIEW koji ima skup trigera koji brišu decu zajedno sa roditeljima.

VIEW ZA SAKRIVANJE SLOŽENE SQL SINTAKSE

Korišćenjem view-ova programeri ne moraju da unose komplikovane SQL naredbe kada žele da dobiju neki rezultat

Korišćenjem VIEW-ova programeri ne moraju da unose komplikovane SQL naredbe kada žele da dobiju neki rezultat.

Primer 9: Pretpostavimo da prodavci iz baze podataka galerije slika žele da vide koji su kupci zainteresovani za koja umetnička dela. Da bi se to pokazalo, potrebno je napraviti JOIN između tri tabele: CUSTOMER, CUSTOMER_ARTIST_INT i ARTIST.

```
CREATE VIEW CustomerInterests AS  
SELECT C.Name Customer, A.Name Artist  
FROM CUSTOMER C, CUSTOMER_ARTIST_INT CI,  
ARTIST A  
WHERE C.CustomerID = CI.CustomerID  
AND A.ArtistID = CI.ArtistID;
```

Ovaj SQL je komplikovan za pisanje, ali kada se kreira **VIEW**, rezultat te naredbe se može dobiti jednostavnom **SELECT** naredbom, što je prikazano na slici 7.6.

```
SELECT *  
FROM CustomerInterests;
```

Customer	Artist
Chris Wilkens	Frings
Chris Wilkens	Tobey
David Smith	Tobey
Donald G. Gray	Tobey
Fred Smathers	Tobey
Lynda Johnson	Tobey
Lynda Johnson	Moos
Lynda Johnson	Frings

Slika 7.6 Rezultat za VIEW - primer 9 [Izvor: NM IT350-2020/2021.]

AŽURIRANJE VIEW-A

Naredba UPDATE VIEW omogućava ažuriranje pogleda. Neki pogledi se mogu ažurirati a drugi ne.

Naredba **UPDATE VIEW** omogućava ažuriranje pogleda. Neki pogledi se mogu ažurirati a drugi ne.

Da bi mogli vršiti ažuriranje pogleda, pogled mora biti definisan nad jednom tabelom, u definiciju pogleda moraju biti uključene sve NOT NULL kolone tabele i kolone pogleda moraju sadržati samo prost, neizmenjen sadržaj kolona tabele nad kojom je pogled definisan.

Primer 10:

```
UPDATE VIEW CustomerTable1  
SET Email = 'NewEmailAddress'  
WHERE CustomerID = 1000;
```

Ažuriranje preko pogleda se ne može izvršiti ako je:

1. pogled definisan nad više tabela. T akvo ažuriranje bi značilo da se jednom naredbom vrši ažuriranje više tabela ba ze podataka, što je suprotno definiciji INSERT, DELETE i UPDATE naredbi.
2. definicija pogleda napravljena tako da se iza SELECT klauzule nalaze funkcije i aritmetički izrazi.
3. definicija pogleda napravljena tako da u nju nisu uključene sve NOT NULL kolone tabele nad kojom je pogled definisan.

Primer 11.

```
UPDATE VIEW WorkNet  
SET TotalNet = 230000  
WHERE Artist = 'Tobey';
```

Ovakvo ažuriranje je **nemoguće** jer je TotalNet suma izračunatih kolona koja se ne nalazi nigde u bazi podataka.

BRISANJE VIEW-A

Naredba DROP VIEW služi za izbacivanje predhodno kreiranog pogleda

Naredba **DROP VIEW** služi za izbacivanje predhodno kreiranog pogleda.

Primer 12:

```
DROP VIEW FINANSIJE;
```

Kako pogled nema svoje podatke, ovom naredbom se samo izbacuje definicija pogleda iz rečnika, a podaci ostaju u bazi podataka.

▼ Poglavlje 8

Pokazna vežba

NAČIN ORGANIZACIJE POKAZNIH VEŽBI

Vežba je organizovana kroz uvod deo i deo za samostalni rad studenata

Vežba je organizovana kroz uvod deo i deo za samostalni rad studenata.

1. U uvodnom delu pokaznih vežbi se daje pokazni primer koji studentima treba da pomogne u samostalnom rešavanju zadataka.
2. Zadatke koji su zadati za samostalni rad student samostalno rešava uz pomoć asistenta.

▼ 8.1 Pokazni primer

PRIKAZ TABELE STUDENTSKE BAZE - DOSIJE (5 MIN.)

Studentska baza - tabela DOSIJE

```
CREATE TABLE IF NOT EXISTS dosije(  
  INDEKS int(11) NOT NULL,  
  IME varchar(50) NOT NULL,  
  PREZIME varchar(50) NOT NULL,  
  GOD_RODJENJA smallint(6) NOT NULL,  
  MESTO_RODJENJA varchar(100) NOT NULL,  
  PRIMARY KEY (INDEKS)  
)  
ENGINE=InnoDB  
DEFAULT CHARSET = latin1;
```

Odgovarajući podaci za datu tabelu.

```
INSERT INTO  
dosije (INDEKS, IME, PREZIME, GOD_RODJENJA, MESTO_RODJENJA)  
VALUES  
(20100021, 'Milos', 'Peric', 1992, 'Beograd'),
```

```
(20100022, 'Marijana', 'Savkovic', 1993, 'Kraljevo'),
(20100023, 'Sanja', 'Terzic', 1991, 'Beograd'),
(20100024, 'Nikola', 'Vukovic', 1992, ''),
(20100025, 'Ljubica', 'Savkovic', 1991, 'Kraljevo'),
(20100026, 'Zorica', 'Miladinovic', 1993, 'Vranje'),
(20100027, 'Milena', 'Stankovic', 0, '');
```

PRIKAZ TABELE STUDENTSKE BAZE - ISPIT (5 MIN.)

Studentska baza - tabela ISPIT

```
CREATE TABLE IF NOT EXISTS ispit(
    indeks int(11) NOT NULL,
    id_predmeta int(11) NOT NULL,
    godina_roka smallint(6) NOT NULL,
    oznaka_roka char(5) NOT NULL,
    ocena smallint(6) NOT NULL,
    datum_ispita DATE NOT NULL,

    PRIMARY KEY (indeks,id_predmeta,godina_roka,oznaka_roka)
)
ENGINE=InnoDB
DEFAULT CHARSET=latin1;
```

Odgovarajući podaci za datu tabelu.

```
INSERT INTO ispit (indeks, id_predmeta,
godina_roka, oznaka_roka, ocena, datum_ispita)
VALUES
(20100021, 1001, 2011, 'jan', 9, '2020-01-20'),
(20100021, 1021, 2011, 'apr', 7, '2003-04-20'),
(20100021, 2001, 2011, 'jan', 10, '2025-01-20'),
(20100021, 3001, 2011, 'jan', 7, '2027-01-20'),
(20100022, 1001, 2011, 'jan', 8, '2020-01-20'),
(20100022, 1021, 2011, 'apr', 5, '2003-04-20'),
(20100022, 2001, 2011, 'jan', 9, '2025-01-20'),
(20100023, 1001, 2011, 'jan', 8, '2020-01-20'),
(20100023, 1021, 2011, 'apr', 10, '2003-04-20'),
(20100023, 2001, 2011, 'jan', 8, '2025-01-20'),
(20100023, 3001, 2011, 'jan', 5, '2027-01-20'),
(20100024, 1001, 2011, 'jan', 10, '2020-01-20'),
(20100024, 1021, 2011, 'apr', 6, '2003-04-20'),
(20100024, 2001, 2011, 'jan', 7, '2025-01-20'),
(20100024, 3001, 2011, 'jan', 6, '2028-01-20'),
(20100025, 1001, 2011, 'jan', 6, '2020-01-20'),
(20100025, 2001, 2011, 'feb', 6, '2010-02-20'),
(20100025, 2001, 2011, 'jan', 5, '2025-01-20'),
(20100026, 1001, 2011, 'feb', 7, '2010-02-20'),
(20100026, 1001, 2011, 'jan', 5, '2020-01-20'),
```

```
(20100026, 1021, 2011, 'apr', 8, '2003-04-20'),
(20100026, 2001, 2011, 'feb', 7, '2010-02-20'),
(20100026, 3001, 2011, 'jan', 6, '2028-01-20');
```

PRIKAZ TABELE STUDENTSKE BAZE - PREDMET (5 MIN.)

Studentska baza - tabela PREDMET

```
CREATE TABLE IF NOT EXISTS predmet (
  id_predmeta int(11) NOT NULL,
  sifra varchar(20) NOT NULL,
  naziv varchar(200) NOT NULL,
  bodovi smallint(6) NOT NULL,
  PRIMARY KEY (id_predmeta)
)
ENGINE=InnoDB
DEFAULT CHARSET=latin1;
```

Odgovarajući podaci za datu tabelu.

```
INSERT INTO predmet (id_predmeta, sifra, naziv, bodovi)
VALUES
(1001, 'M111', 'Analiza 1', 6),
(1002, 'M112', 'Analiza 2', 6),
(1003, 'M113', 'Analiza 3', 6),
(1021, 'M131', 'Geometrija', 6),
(1101, 'M105', 'Diskretne strukture 1', 6),
(1102, 'M106', 'Diskretne strukture 2', 6),
(2001, 'P101', 'Programiranje 1', 8),
(2002, 'P102', 'Programiranje 2', 8),
(2003, 'P103', 'Objektno orijentisano programiranje', 6),
(2004, 'P104', 'Algoritmi i strukture podataka', 6),
(3001, 'S1', 'Engleski jezik 1', 5),
(3002, 'S2', 'Engleski jezik 2', 5),
(4001, 'R101', 'Uvod u organizaciju racunara', 5),
(4002, 'R102', 'Uvod u Veb i Internet tehnologije', 5);
```

PRIKAZ TABELE STUDENTSKE BAZE - ISPITNI_ROK (5 MIN.)

Studentska baza - tabela ISPITNI_ROK

```
CREATE TABLE IF NOT EXISTS ispitni_rok (
  godina_roka smallint(6) NOT NULL,
  oznaka_roka varchar(20) NOT NULL,
  naziv varchar(50) NOT NULL,
  PRIMARY KEY (godina_roka, oznaka_roka)
)
```

```
ENGINE=InnoDB
DEFAULT CHARSET=latin1;
```

Odgovarajući podaci za datu tabelu.

```
INSERT INTO ispitni_rok (godina_roka, oznaka_roka, naziv)
VALUES
(2011, 'apr', 'April 2011'),
(2011, 'feb', 'Februar 2011'),
(2011, 'jan', 'Januar 2011'),
(2011, 'jun', 'Jun 2011'),
(2011, 'okt', 'Oktobar 2011'),
(2011, 'sep', 'Septembar 2011');
```

PRIMERI 1. PODUPITI ZA SPAJANJE TABELA (9 MIN.)

Naredni primer služi za utvrđivanje pisanja podupita

Primer 1. Pronaći sva imena i prezimena studenata iz tabele dosije koji žive u gradu u kome živi i student čiji je broj indeksa 20100021.

```
SELECT ime, prezime
FROM dosije
WHERE mesto_rodjenja = (
    SELECT mesto_rodjenja FROM dosije
    WHERE indeks = 20100021
);
```

Osim na prikazani način, moguće je još uz pomoć IN klazule dobiti iste rezultate.

```
SELECT ime, prezime
FROM dosije
WHERE mesto_rodjenja IN (
    SELECT mesto_rodjenja FROM dosije
    WHERE indeks = 20100021
);
```

Iz navedenih upita dobijamo rezultat prikazan na slici 9.1:

	ime	prezime
►	Milos	Peric
	Sanja	Terzic

Slika 8.1.1 Izlaz iz SELECT naredbe, primer 1 [Izvor: NM IT350-2020/2021.]

Primer 2. Izdvojiti imena i prezimena studenata koji su položili predmet čiji je ID = 2001.

```
SELECT ime, prezime  
FROM dosije  
WHERE indeks IN (  
    SELECT indeks  
    FROM ispit  
    WHERE id_predmeta = 2001 AND ocena > 5  
);
```

Iz navedenog upita dobijamo rezultat prikazan na slici 9.2:

	ime	prezime
►	Milos	Peric
	Marjana	Savkovic
	Sanja	Terzic
	Nikola	Vukovic
	Ljubica	Savkovic
	Zorica	Miladinovic

Slika 8.1.2 Izlaz iz SELECT naredbe, primer 2 [Izvor: NM IT350-2020/2021.]

PRIMERI 2. PODUPITI ZA SPAJANJE TABELA (7 MIN.)

Cilj primera jeste korišćenje operatora EXISTS i NOT u podupitu

Primer 3. Izdvojiti imena i prezimena studenata koji su položili predmet čiji je ID = 2001. Koristiti operator EXISTS.

```
SELECT ime, prezime  
FROM dosije as D  
WHERE EXISTS (  
    SELECT *  
    FROM ispit  
    WHERE indeks = D.indeks  
    AND id_predmeta = 2001  
);
```

Iz navedenog upita dobijamo rezultat prikazan na slici 9.3:

	ime	prezime
►	Milos	Peric
	Manjana	Savkovic
	Sanja	Terzic
	Nikola	Vukovic
	Ljubica	Savkovic
	Zorica	Miladinovic

Slika 8.1.3 Izlaz iz SELECT naredbe, primer 3 [Izvor: NM IT350-2020/2021.]

Primer 4. Izdvojiti imena i prezimena studenata koji nisu položili predmet čiji je ID = 2001. Koristiti operatore NOT i EXISTS.

```
SELECT ime, prezime
FROM dosije as D
WHERE NOT EXISTS (
    SELECT *
    FROM ispit
    WHERE indeks = D.indeks
    AND id_predmeta = 2001
);
```

Iz navedenog upita dobijamo izraz prikazan na slici 9.4:

	ime	prezime
►	Milena	Stankovic

Slika 8.1.4 Izlaz iz SELECT naredbe, primer 4 [Izvor: NM IT350-2020/2021.]

PRIMERI 3. PODUPITI ZA SPAJANJE TABELA (5 MIN.)

Vežbanje pisanja podupita sa primenom IN operatora

Primer 5. Izdvojiti brojeve indeksa, imena i prezimena studenta koji su polagali predmet koji nosi 5 bodova.

```
SELECT indeks, ime, prezime
FROM dosije
WHERE indeks IN (
    SELECT indeks
    FROM ispit
    WHERE bodovi = 5
);
```



```
WHERE id_predmeta IN (
    SELECT id_predmeta
    FROM predmet
    WHERE bodovi = 5
)
);
```

Iz navedenog upita dobijamo izlaz prikazan na slici 9.5:

	indeks	ime	prezime
►	20100021	Milos	Peric
	20100023	Sanja	Terzic
	20100024	Nikola	Vukovic
	20100026	Zorica	Miladinovic

Slika 8.1.5 Izlaz iz SELECT naredbe, primer 5 [Izvor: NM IT250-2020/2021.]

PRIMERI 1-2 KORIŠĆENJA JOIN OPERATORA (7 MIN.)

Primeri korišćenja JOIN operatora

Primer 1. Prikazati ime i prezime svih studenata koji su položili bar jedan predmet.

```
SELECT DISTINCT ime, prezime
FROM dosije
JOIN ispit ON ispit.indeks = dosije.indeks;
```

Iz navedenih upita dobijamo izlaz prikazan na slici 9.6:

	ime	prezime
►	Milos	Peric
	Manjana	Savkovic
	Sanja	Terzic
	Nikola	Vukovic
	Ljubica	Savkovic
	Zorica	Miladinovic

Slika 8.1.6 Izlaz iz SELECT naredbe, primer 1 [Izvor: NM IT350-2020/2021.]

Primer 2. Prikazati ime i prezime svih studenata bez obzira da li su položili neki predmet. Potrebno je koristiti LEFT JOIN operator.

```
SELECT DISTINCT ime, prezime  
FROM dosije  
LEFT JOIN ispit ON ispit.indeks = dosije.indeks;
```

Iz navedenih upita dobijamo izlaz prikazan na slici 9.7:

	ime	prezime
►	Milos	Peric
	Marjana	Savkovic
	Sanja	Terzic
	Nikola	Vukovic
	Ljubica	Savkovic
	Zorica	Miladinovic
	Milena	Stankovic

Slika 8.1.7 Izlaz iz SELECT naredbe, primer 2 [Izvor: NM IT350-2020/2021.]

PRIMER 3. KORIŠĆENJE JOIN OPERATORA (7 MIN.)

Primeri korišćenja JOIN operatora (2)

Primer 3. Izdvojiti broj indeksa i nazive svih predmeta koje je student polagao 2011 godine. Rezultat urediti prema broju indeksa, rastuće.

```
SELECT indeks, naziv  
FROM ispit AS I  
JOIN predmet AS P  
ON I.id_predmeta = P.id_predmeta  
WHERE i.godina_roka = 2011  
ORDER BY indeks;
```

Iz navedenog upita dobijamo izlaz prikazan na slici 9.8:

	indeks	naziv
►	20100021	Analiza 1
	20100021	Geometrija
	20100021	Programiranje 1
	20100021	Engleski jezik 1
	20100022	Analiza 1
	20100022	Geometrija
	20100022	Programiranje 1
	20100023	Analiza 1
	20100023	Geometrija
	20100023	Programiranje 1
	20100023	Engleski jezik 1
	20100024	Analiza 1
	20100024	Geometrija
	20100024	Programiranje 1
	20100024	Engleski jezik 1
	20100025	Analiza 1
	20100025	Programiranje 1
	20100025	Programiranje 1
	20100026	Analiza 1
	20100026	Analiza 1
	20100026	Geometrija
	20100026	Programiranje 1
	20100026	Engleski jezik 1

Slika 8.1.8 Izlaz iz SELECT naredbe, primer 3 [Izvor: NM IT350-2020/2021.]

PRIMERI 4-5 KORIŠĆENJA JOIN OPERATORA (8 MIN.)

Primer spajanja tri tabele JOIN operatorima. Primer sa agregatnim funkcijama

Primer 4. Izdvojiti brojeve indeksa, imena i prezimena studenta koji su polagali predmet koji nosi 5 bodova.

```
SELECT D.indeks, D.ime, D.prezime
FROM dosije D
JOIN ispit I
ON D.indeks = I.indeks
JOIN predmet P
ON I.id_predmeta = p.id_predmeta
WHERE P.bodovi = 5;
```

Iz navedenih upita dobijamo izlaz prikazan na slici 9.9:

	indeks	ime	prezime
►	20100021	Milos	Peric
	20100023	Sanja	Terzic
	20100024	Nikola	Vukovic
	20100026	Zorica	Miladinovic

Slika 8.1.9 Izlaz iz SELECT naredbe, primer 4 [Izvor: NM IT350-2020/2021.]

Primer 5: Za svakog studenta koji je skupio barem 20 bodova prikazati ukupan broj skupljenih bodova.

```
SELECT indeks, SUM(bodovi) AS Ukupno
FROM ispit I
JOIN predmet P
ON I.id_predmeta = P.id_predmeta
WHERE ocena > 5
GROUP BY indeks
HAVING SUM(bodovi) >= 20;
```

Iz navedenih upita dobijamo izlaz prikazan na slici 9.10:



	indeks	Ukupno
►	20100021	25
	20100023	20
	20100024	25
	20100026	25

Slika 8.1.10 Izlaz iz SELECT naredbe, primer 5 [Izvor: NM IT350-2020/2021.]

PRIMERI: JOIN (5 MIN.)

JOIN tabela CUSTOMER, ORDER, ORDERITEM i PRODUCT

Date su tabele ORDER i CUSTOMER prikazane na slici 9.11.

ORDER	CUSTOMER
Id 	Id 
OrderDate	FirstName
OrderNumber	LastName
CustomerId	City
TotalAmount	Country
	Phone

Slika 8.1.11 Tabele ORDER i CUSTOMER [Izvor: NM IT250-2020/2021.]

Zadatak 1: Izlistaj sve narudžbine sa informacijama o kupcima

```
SELECT OrderNumber, TotalAmount, FirstName, LastName, City, Country
FROM Order JOIN Customer
ON Order.CustomerId = Customer.Id
```

OrderNumber	TotalAmount	FirstName	LastName	City	Country
542378	440.00	Paul	Henriot	Reims	France
542379	1863.40	Karin	Josephs	Münster	Germany
542380	1813.00	Mario	Pontes	Rio de Janeiro	Brazil
542381	670.80	Mary	Saveley	Lyon	France
542382	3730.00	Pascale	Cartrain	Charleroi	Belgium
542383	1444.80	Mario	Pontes	Rio de Janeiro	Brazil
542384	625.20	Yang	Wang	Bern	Switzerland
...					

Slika 8.1.12 Izlaz iz naredbe SELECT iz zadatka 1. [Izvor: NM IT350-2020/2021.]

Zadatak 2: Analizirati rezultat naredbe SELECT.

```
SELECT O.OrderNumber, CONVERT(date,O.OrderDate) AS Date,
       P.ProductName, I.Quantity, I.UnitPrice
FROM [Order] O
JOIN OrderItem I ON O.Id = I.OrderId
JOIN Product P ON P.Id = I.ProductId
ORDER BY O.OrderNumber
```

U ovom upitu se izvršavaju dve JOIN operacije sa 3 tabele. O, I i P su alijasi tabela. Date je alijas kolone. Dobija se spisak svih narudžbina sa imenima proizvoda, količinama i cenama na osnovu tri tabele.

PRIMERI: LEFT JOIN I RIGHT JOIN (4 MIN.)

LEFT JOIN i RIGHT JOIN tabela ORDER i CUSTOMER

```
SELECT OrderNumber, TotalAmount, FirstName, LastName, City, Country
FROM Customer C LEFT JOIN [Order] O
```

```
ON 0.CustomerId = C.Id
ORDER BY TotalAmount
```

LEFT JOIN: Upitom će biti izlistani svi kupci bez obzira da li su oni plasirali ili nisu plasirali porudžbinu. Izlaz iz upita dat je slici 9.13.

OrderNumber	TotalAmount	FirstName	LastName	City	Country
NULL	NULL	Diego	Roel	Madrid	Spain
NULL	NULL	Marie	Bertrand	Paris	France
542912	12.50	Patricio	Simpson	Buenos Aires	Argentina
542937	18.40	Paolo	Accorti	Torino	Italy
542897	28.00	Pascale	Cartrain	Charleroi	Belgium
542716	28.00	Maurizio	Moroni	Reggio Emilia	Italy
543028	30.00	Yvonne	Moncada	Buenos Aires	Argentina
543013	36.00	Fran	Wilson	Portland	USA

Slika 8.1.13 Izlaz koji se dobija iz prethodnog SELECT-a sa LEFT JOIN-om [Izvor: NM IT350-2020/2021.]

```
SELECT TotalAmount, FirstName, LastName, City, Country
FROM [Order] O RIGHT JOIN Customer C
ON O.CustomerId = C.Id
WHERE TotalAmount IS NULL
```

RIGHT JOIN: Upitom će se dobiti kupci koji kada se spoje, nemaju odgovarajuću porudžbinu.

TotalAmount	FirstName	LastName	City	Country
NULL	Diego	Roel	Madrid	Spain
NULL	Marie	Bertrand	Paris	France

Slika 8.1.14 Izlaz koji se dobija iz prethodnog SELECT-a sa RIGHT JOIN-om [Izvor: NM IT350-2020/2021.]

PRIMER: FULL JOIN (5 MIN.)

FULL JOIN tabela CUSTOMER i SUPPLIER

Na sici su prikazane tabele CUSTOMER i SUPPLIER

CUSTOMER	SUPPLIER
Id 🔑	Id 🔑
FirstName	CompanyName
LastName	ContactName
City	City
Country	Country
Phone	Phone
	Fax

Slika 8.1.15 Tabele CUSTOMER i SUPPLIER [Izvor: NM IT350-2020/2021.]

Uporedi sve kupce i dobavljače po zemljama

```
SELECT C.FirstName, C.LastName, C.Country AS CustomerCountry,
       S.Country AS SupplierCountry, S.CompanyName
FROM Customer C FULL JOIN Supplier S
ON C.Country = S.Country
ORDER BY C.Country, S.Country
```

SELECT vraća dobavljače koji nemaju kupce u svojim zemljama i kupce koji nemaju dobavljače u svojim zemljama, i dobavljače i kupce koji su iz iste zemlje.

NULL	NULL	NULL	Japan	Tokyo Traders
NULL	NULL	NULL	Japan	Mayumi's
NULL	NULL	NULL	Netherlands	Zaanse Snoepfabriek
NULL	NULL	NULL	Singapore	Leka Trading
Patricio	Simpson	Argentina	NULL	NULL
Yvonne	Moncada	Argentina	NULL	NULL
Sergio	Gutiérrez	Argentina	NULL	NULL
Georg	Pipps	Austria	NULL	NULL
Roland	Mendel	Austria	NULL	NULL
Pascale	Cartrain	Belgium	NULL	NULL
Catherine	Dewey	Belgium	NULL	NULL
Bernardo	Batista	Brazil	Brazil	Refrescos Americanas LTDA
Lúcia	Carvalho	Brazil	Brazil	Refrescos Americanas LTDA

Slika 8.1.16 Izlaz koji se dobija iz prethodnog SELECT-a sa FULL JOIN-om [Izvor: NM IT350-2020/2021.]

PRIMERI 1-2 ZA KREIRANJE POGLEDA (8 MIN.)

Primer kreiranja pogleda

Primer 1. Kreirati pogled kojim se izdvajaju svi ispitni rokovi počev od 2008. godine.

```
CREATE VIEW ispitniRok2010 AS
SELECT godina_roka, oznaka_roka, naziv
FROM ispitni_rok
WHERE godina_roka >= 2010;
```

Na ovaj način kreirali smo novi VIEW sa podacima iz navedene tabele. Da bi smo videli koje podatke sadrži novo nastala (virtualna) tabela moramo da izvršimo SELECT upit na njom.

```
SELECT *
FROM ispitnirok2010;
```

Iz navedenih upita dobijamo izlaz prikazan na slici 9.17.

	godina_roka	oznaka_roka	naziv
▶	2011	apr	April 2011
	2011	feb	Februar 2011
	2011	jan	Januar 2011
	2011	jun	Jun 2011
	2011	okt	Oktobar 2011
	2011	sep	Septembar 2011

Slika 8.1.17 Izlaz iz SELECT naredbe, primer 1 [Izvor: NM IT250-2020/2021.]

Primer 2. Kreirati pogled kojim se izdvajaju sva polaganja studenata koji se zovu Marko ili Maja, a u aprilskom ispitnom roku 2008. godine su položili barem jedan predmet.

```
CREATE VIEW polaganja_stud_kv_bg1 AS
SELECT I1.*
FROM dosije D
JOIN ispit I1 ON D.indeks = I1.indeks
JOIN ispit I2 ON I2.indeks = I1.indeks
WHERE ime IN ('Marko', 'Maja')
AND I2.oznaka_roka = 'apr'
AND I2.godina_roka >= 2008
AND I2.ocena > 5;
```

Vaš zadatak je da ovaj primer ispitajte da li je ispravan i ukoliko je potrebno, uradite potrebne korekcije ili na primeru ili na samoj bazi.

PRIMERI 3-4 ZA AŽURIRANJE I BRISANJE POGLEDA (5 MIN.)

Primer za izmenu podataka u pogledu. Brisanje pogleda

Primer 3. Kreirati pogled koji ispisuje sva imena i prezimena svih studenata na fakultetu. Potom promeniti prezime svim studentkinjama koje se zovu Sanja u Arsic.

```
CREATE VIEW pogled AS
SELECT ime, prezime
FROM dosije;
```

Na ovaj način kreirali smo novi VIEW sa podacima iz navedene tabele. Da bi smo videli koje podatke sadrži novo nastala (virtualna) tabela moramo da izvršimo SELECT upit na njom.

```
SELECT * FROM pogled;
```

Iz navedenih upita dobijamo izlaz prikazan na slici 9.18:

	ime	prezime
►	Milos	Peric
	Manjana	Savkovic
	Sanja	Terzic
	Nikola	Vukovic
	Ljubica	Savkovic
	Zorica	Miladinovic
	Milena	Stankovic

Slika 8.1.18 Izlaz iz SELECT naredbe, primer 3 [Izvor: NM IT250-2020/2021.]

```
UPDATE pogled
SET prezime = 'Arsic'
WHERE ime = 'Sanja';
```

Nakon ažuriranja pogleda primenom naredbe SELECT

```
SELECT * FROM pogled
WHERE ime = 'Sanja';
```

dobijamo sadržaj pogleda prikazan na slici 9.19:

	Sanja	Arsic
--	-------	-------

Slika 8.1.19 Izlaz iz SELECT naredbe nakon ažuriranja pogleda, primer 3 [Izvor: NM IT350-2020/2021.]

Primer 4. Obrisati prethodno kreiran pogled.

```
DROP VIEW pogled;
```

▼ 8.2 Zadaci za samostalni rad

GRUPA 1 ZADATAKA ZA SAMOSTALNI RAD (30 MIN.)

Cilj narednih primera jeste da student samostalno uradi zadate upite

Nad bazom podataka o studentima uraditi sledeće zadatke:

1. Zadatak 1. Za svaki od ispitnih rokova i za svaki polagan predmet odrediti broj položenih ispita.

2. Zadatak 2. Napisati upit u SQL-u kojim se za sve studente upisane 2010. godine na fakultet, čiji je prosek barem 7.5 ispisuje pored imena i prezimena i 'vrhunski' ako ima prosek preko 9, 'odličan' ako ima prosek preko 8, odnosno 'dobar' ako mu je prosek između 7.5 i 8.
3. Zadatak 3. Kreirati pogled koji za svakog od studenata čija je prosečna ocena veća od 8, izdvojiti broj indeksa i ukupan broj sakupljenih bodova.
4. Zadatak 4. Prikazati ukupan broj položenih ispita studenta sa brojem indeksa 20100022, kolonu preimenovati u "Ukupno položeno".
5. Zadatak 5. Kreirati pogled koji izdvaja nazive ispitnih rokova u kojima je prosečna ocena veća od 7 i u kojima je više od 2 studenta položilo neki (bilo koji) ispit.
6. Zadatak 6. Izdvojiti brojeve indeksa studenata koji su polagali (no mora da znači da su i položili) predmet koji nosi 5 kredita.

Potrebno vreme za rešavanje zadataka 30 minuta

GRUPA 2 ZADATAKA ZA SAMOSTALNI RAD (108 MIN.)

Cilj naredne sekcije jeste da student samostalno uradi zadate upite

1. Napisati upit kojim se prikazuju podaci o svim studentima koji su položili predmet CS101 I imaju prosečnu ocenu veću od 8,5.
2. Napisati upit kojim se prikazuju podaci o svim profesorima koji predaju više od tri predmeta.
3. Napisati upit kojim se prikazuju podaci o 4 najuspešnijih predmeta, tj. Predmeta koja je položio najveći broj studenata.
4. Napisati upit kojim se prikazuju podaci o smeru na kome ima najviše studenata koji su položili Engleski 1.
5. Napisati upit kojim se za svaki predmet koji ima više od 5 časova predavanja I vežbi zajedno prikazuje broj studenata koji su taj predmet položili u koloni koja je imenovana sa "položili", zatim ukupan broj studenata koji je taj predmet pao u koloni koja se zove "pali" I u poslednjoj koloni prikazati prosečnu ocenu na tom predmetu I kolonu imenovati sa "prosečna ocena".
6. Prikazati podatke o predmetu I studentu čija je ocena upisana poslednja u tabelu overa. Poslednja ocena ima najveći datum u tabeli overa.
7. Napisati upit kojim se prikazuje prosečna ocena za sve student koji su rođeni u 8 ili 2. Mesecu.
8. Napisati upit kojim se za sve student čiji broj telefona počinje sa 062 I koji su tradicionalni student prikazuje broj predmeta koji su položili.
9. Napisati upit kojim se prikazuju broj indeksa, imena I prezimena svih studenata koji su položili bar jedan predmet kao I student sa brojem indeksa 1568.
10. Napisati upit kojim se prikazuju podaci o profesorima koji predaju predmete sa 8espb-a.
11. Napisati upit kojim se prikazuje za sve predavače na fakultetu čije je zvanje Profesor koliko predmeta predaju.
12. Napisati upit kojim se za svakog profesora računa njegov fond časova na nedeljnom nivou. Potrebno je sabrati sve časove predavanja za sve predmete koje taj profesor predaje.
13. Napisati upit kojim se za svakog asistenta ili saradnika računa njegov fond časova na nedeljnom nivou. Potrebno je sabrati sve časove vežbi za sve predmete koje taj profesor

predaje.

14. Za sve studente rođene 1995 godine prikazati indeks, ime, prezime, naziv predmeta, ocenu i datum kada su ocenu dobili.

15. Za sve student čija je prosečna ocena veća od 7,5 prikazati indeks, ime i prezime, prosečnu ocenu.

16. Prikazati nazive predmeta i broj espb-a za one predmete koje je položilo manje od 5 studenta.

17. Napisati upit kojim se za svakog profesora i svaki predmet koji on predaje računa prosečna ocena za student koji su taj predmet položili.

18. Napisati upit kojim svakom profesoru računa plata na osnovu fond časova, ako se zna da semestar ima 15 nedelja i zna se broj časova predavanja u svakoj nedelji za svaki predmet koji taj profesor drži. Pretpostaviti da se jedan čas profesoru plaća 2000 dinara.

PREOSTALI ZADACI IZ GRUPE 2 ZADATAKA ZA SAMOSTALNI RAD (12 MIN.)

Cilj sekcije jeste da student samostalno uradi zadate upite

19. Napisati upit koji svakom studentu računa koliko novca treba da uplati na račun fakultetata zbog polaganja ispita. Za svakog studenta treba izračunati koliko je puta polagao ispite. Neka cena jednog polaganja bude 1500 dinara.

20. Napisati upit kojim se svim tradicionalnim studentima smera SI računa prosečna ocena. Prikazati samo one student čija je prosečna ocena veća od 6,5 a manja od 9.

Potrebno vreme za rešavanje zadataka 120 minuta (deo zadataka treba rešiti kod kuće)

▼ Poglavlje 9

Domaći zadatak

TEKST ZADATKA 1 - 15. - VREME IZRADE NA OSNOVU UPUTSTVA JE 75 MIN.

Postavka zadatka 1 - 15.

Nad studentskom bazom podataka postaviti i izvršiti sledeće upite:

1. Napisati upit kojim se prikazuju podaci o studentima koji su upisali smer Softversko inženjerstvo. Ispisati broj indeksa, ime, prezime i način studiranja.
2. Napisati upit kojim se prikazuju podaci o svim studentima koji su nekad dobili ocenu 5.
3. Napisati upit kojim se prikazuju podaci o profesoru koji predaje predmet MA101. Prikazati ime i prezime profesora.
4. Napisati upit kojim se prikazuju podaci o studentima koji su na smeru Informacione tehnologije, a koji nisu rođeni u Beogradu.
5. Napisati upit kojim se prikazuju podaci o svim studentima koji su položili predmet CS220. Potrebno je prikazati broj indeksa, ime, prezime i ocenu za svakog studenta. Student je položio predmet ako je dobio veću ocenu od 5.
6. Napisati upit kojim se prikazuje prosečna ocena za svakog studenta na Fakultetu. U prosečnu ocenu se ne računaju ocene 5, već samo položeni predmeti.
7. Napisati upit kojim se prikazuje za svaki od predmeta na fakultetu koliko studenata je položilo isti. Prikazati id predmeta, naziv predmeta i broj studenata koji su položili predmet.
8. Napisati upit kojim se prikazuje za svaki predmet ukupan broj polaganja, bilo da je student pao ili položio. Prikazati id predmeta, naziv predmeta, ukupan broj časova (i predavanja i vežbi) kao i broj svih studenata koji su predmet polagali, ne mora da znači da su i položili.
9. Napisati upit kojim se prikazuje prosečna ocena na svakom predmetu na fakultetu. Prikazati id predmeta, naziv i prosečnu ocenu koju su dobili svi studenti koji su predmet položili. Ne računati one koji su predmet pali.
10. Napisati upit kojim se prikazuje za svaki od smerova id smer, naziv i ukupan broj studenata na tom smeru.
11. Napisati upit kojim se prikazuju podaci o studentu koji ima najbolji prosek na fakultetu. Prikazati sve podatke o najboljem.
12. Napisati upit kojim se prikazuju podaci o predmetu koji je položilo najviše studenata.
13. Napisati upit kojim se prikazuje za svakog profesora na fakultetu broj predmeta koje predaje. Prikazati ime i prezime profesora i broj predmeta.
14. Napisati upit kojim se prikazuje za svakog studenta koliko je ispita položio. Prikazati ime, prezime i broj položenih predmeta.
15. Napisati upit kojim se prikazuju podaci o predmetima koji je položilo manje od 3 studenata.

TEKST ZADATKA 16 - 30. - VREME IZRADE NA OSNOVU UPUTSTVA JE 75 MIN.

Postavka zadatka 16 - 30.

16. Napisati upit kojim se prikazuju podaci o svim studentima koji su položili predmet IT101 i imaju prosečnu ocenu veću od 7,5.
17. Napisati upit kojim se prikazuju podaci o svim profesorima koji predaju dva ili tri predmeta.
18. Napisati upit kojim se prikazuju podaci o 4 najneuspešnijim predmetima, tj. predmetima koja je položio najmanji broj studenata.
19. Napisati upit kojim se prikazuju podaci o smeru na kome ima najmanje studenata koji su položili Engleski 1.
20. Napisati upit kojim se za svaki predmet koji ima više od 3 časova predavanja i vežbi zajedno prikazuje broj studenata koji su taj predmet položili u koloni koja je imenovana sa "polažili", zatim ukupan broj studenata koji je taj predmet pao u koloni koja se zove "pali" i u poslednjoj koloni prikazati prosečnu ocenu na tom predmetu i kolonu imenovati sa "prosečna ocena".
21. Prikazati podatke o predmetu i studentu čija je ocena upisana prva u tabelu overa. Prva ocena ima najmanji datum u tabeli overa.
22. Napisati upit kojim se prikazuje prosečna ocena za sve studente koji su rođeni u 2, 4, 6, 8, 10 ili 12. mesecu.
23. Napisati upit kojim se za sve studente čiji broj telefona počinje sa 062 ili 063 i koji su tradicionalni studenti prikazuje broj predmeta koji su nisu položili.
24. Napisati upit kojim se prikazuju broj indeksa, imena i prezimena svih studenata koji su položili bar dva predmeta kao i student sa brojem indeksa 1568.
25. Napisati upit kojim se prikazuju podaci o profesorima koji predaju predmete sa 6, 7 i 8 espb-a.
26. Napisati upit kojim se za svakog profesora računa njegov fond časova na mesečnom nivou. Potrebno je sabrati sve časove predavanja za sve predmete koje taj profesor predaje. Jedan mesec sadrži 4.2 radne nedelje.
27. Napisati upit kojim se za svakog asistenta ili saradnika računa njegov fond časova na mesečnom nivou. Potrebno je sabrati sve časove vežbi za sve predmete koje taj profesor predaje. Jedan mesec sadrži 4.2 radne nedelje.
28. Prikazati nazive predmeta i broj espb-a za one predmete koje je položilo više ili tačno 3 studenta.
29. Napisati upit kojim svakom profesoru računa plata na osnovu fonda časova, ako se zna da semestar ima 15 nedelja i zna se broj časova predavanja u svakoj nedelji za svaki predmet koji taj profesor drži. Pretpostaviti da se jedan čas profesoru plaća 3500 dinara.
30. Napisati upit kojim se svim tradicionalnim studentima smera SI ili IT računa prosečna ocena. Prikazati samo one studente čija je prosečna ocena veća od 7,5 a manja od 8,5.

UPUTSTVO ZA IZRADU ZADATAKA

Rešiti zadatak prema postavljenom tekstu

Svaki student radi 3 zadatka. Redni brojevi zadatka se računaju po formulama:

1. zadatak = $(\text{Broj_indeksa} \bmod 15) + 1$
2. zadatak = $((\text{Broj_indeksa} + \text{Redni broj zadatka } 1) \bmod 15) + 1$
3. zadatak = $((\text{Broj_indeksa} + \text{Redni broj zadatka } 2) \bmod 15) + 1$

Rešeni zadatak šaljete kao SQL file (arhiviran) na mail adresu asistenta.

Prilikom slanja domaćih zadatka, neophodno je da ispunite sledeće:

- Subject mail-a mora biti IT250-DZbr (u slučaju kada šaljete domaći za ovu nedelju to je IT250-DZ10)
- U prilogu mail-a treba da se nalazi arhiviran projekat koji se ocenjuje imenovan na sledeći način: IT250-DZbr-BrojIndeksa-Ime Prezime. Na primer, IT250-DZ10-1234-VeljkoGrkovic
- Telo mail-a treba da ima pozdravnu poruku
- Arhivu sa zadatkom poslati na adresu predmetnog asistenta:
milica.vlajkovic@metropolitan.ac.rs (studenti u Beogradu i online studenti) ili
tamara.vukadinovic@metropolitan.ac.rs (studenti u Nišu).

Svi poslati mail-ovi koji ne ispunjavaju navedene uslove NEĆE biti pregledani. Za sva pitanja ili nedoumice u vezi zadatka, možete se obratiti asistentu

▼ Zaključak

ZAKLJUČAK

Šta smo naučili u ovoj lekciji?

U ovoj lekciji je kroz veliki broj primera prikazan način spajanja više tabela.

Posebno su objašnjeni primeri za spajanje tabela Z. ko rišćenjem podupita, naznačena ograničenja koja postoje u primeni ovakvog načina spajanja i posebno prikazani primeri za primenu različitih vrsta JOIN-a kao što su:

1.

JOIN (INNER JOIN,CROSS JOIN, CARTESIAN JOIN), LEFT JOIN (LEFT OUTER JOIN),

2.

RIGHT JOIN (RIGHT OUTER JOIN),

3.

FULL JOIN.

4.

SELF JOIN

Takođe se govorilo i o pogledima (VIEW) koji predstavljaju virtualne tabele koje se primenjuju kada želimo da:

1. sakrijemo neke kolone ili redove tabele

2. prikažemo rezultate nekih izračunavanja

3. sakrijemo složenu SQL sintaksu

4. slojevito ugradimo neke funkcije

5. omogućimo nivo izolacije između podataka tabele i korisnikovog pogleda na podatke

6. nad istom tabelom različitim korisnicima dodelimo različite dozvole

7. nad istom tabelom različitim view-ovima dodelimo različite trigere

LITERATURA

Za pisanje ove lekcije korišćena je sledeća literatura:

1. C. J. Date, An introduction to Database Systems, Addison-Wesley Publishing Company, 1990

2. https://www.w3schools.com/SQL/sql_join.asp

3. https://www.w3schools.com/SQL/sql_join_inner.asp

4. https://www.w3schools.com/SQL/sql_join_left.asp

5. https://www.w3schools.com/SQL/sql_join_right.asp
6. https://www.w3schools.com/SQL/sql_join_full.asp
7. https://www.w3schools.com/SQL/sql_join_self.asp
8. https://www.w3schools.com/SQL/sql_alias.asp
9. <https://www.w3resource.com/sql/subqueries/understanding-sql-subqueries.php>