MySQL - Analiza i optimizacija upita

Evidencija račun

Potrebno je evidentirati račune izdane u trgovini. Za svaki račun potrebno je pratiti *broj_računa*, *datum_izdavanja* i *zaposlenika* koji je račun izdao, te kupca koji je platio račun. Za zaposlenika se prati: *ime*, *prezime*, *oib*, *datum_zaposlenja*, dok se za kupca prati *ime* i *prezime*. Na svakom računu se nalazi stavke koje u sebi sadrže *artikl* i *količinu*. Artikl se sastoji od *naziva* i *cijene*.

```
kupac(id, ime, prezime)
zaposlenik(id, ime, prezime, oib, datum_zaposlenja)
artikl(id, naziv, cijena)
racun(id, id_zaposlenik, id_kupac, broj, datum_izdavanja)
stavka_racun(id, id_racun, id_artikl, kolicina)
```

Baza podataka & tablice

```
CREATE DATABASE trgovina;
USE trgovina;

CREATE TABLE kupac (
  id INTEGER NOT NULL,
  ime VARCHAR(10) NOT NULL,
  prezime VARCHAR(15) NOT NULL,
  PRIMARY KEY (id)
);

CREATE TABLE zaposlenik (
  id INTEGER NOT NULL,
  ime VARCHAR(10) NOT NULL,
  prezime VARCHAR(15) NOT NULL,
  oib CHAR(11) NOT NULL,
  datum_zaposlenja DATETIME NOT NULL,
  PRIMARY KEY (id)
);

CREATE TABLE artikl (
  id INTEGER NOT NULL,
  naziv VARCHAR(20) NOT NULL,
  cijena NUMERIC(10,2) NOT NULL,
  PRIMARY KEY (id)
);
```

```sql
CREATE TABLE racun (
  id INTEGER NOT NULL,
  id_zaposlenik INTEGER NOT NULL,
  id_kupac INTEGER NOT NULL,
  broj VARCHAR(100) NOT NULL,
  datum_izdavanja DATETIME NOT NULL,
  PRIMARY KEY (id),
  FOREIGN KEY (id_zaposlenik) REFERENCES zaposlenik (id),
  FOREIGN KEY (id_kupac) REFERENCES kupac (id)
);

CREATE TABLE stavka_racun (
  id INTEGER NOT NULL,
  id_racun INTEGER NOT NULL,
  id_artikl INTEGER NOT NULL,
  kolicina INTEGER NOT NULL,
  PRIMARY KEY (id),
  FOREIGN KEY (id_racun) REFERENCES racun (id) ON DELETE CASCADE,
  FOREIGN KEY (id_artikl) REFERENCES artikl (id),
  UNIQUE (id_racun, id_artikl)
);
```

Unos podataka

```sql
INSERT INTO kupac VALUES (1, 'Lea', 'Fabris'),
                         (2, 'David', 'Sirotić'),
                         (3, 'Tea', 'Bibić');

INSERT INTO zaposlenik VALUES
  (11, 'Marko', 'Marić', '123451', STR_TO_DATE('01.10.2020.', '%d.%m.%Y.')),
  (12, 'Toni', 'Milovan', '123452', STR_TO_DATE('02.10.2020.', '%d.%m.%Y.')),
  (13, 'Tea', 'Marić', '123453', STR_TO_DATE('02.10.2020.', '%d.%m.%Y.'));

INSERT INTO artikl VALUES (21, 'Puding', 5.99),
                          (22, 'Milka čokolada', 30.00),
                          (23, 'Čips', 9);

INSERT INTO racun VALUES
  (31, 11, 1, '00001', STR_TO_DATE('05.10.2020.', '%d.%m.%Y.')),
  (32, 12, 2, '00002', STR_TO_DATE('06.10.2020.', '%d.%m.%Y.')),
  (33, 12, 1, '00003', STR_TO_DATE('06.10.2020.', '%d.%m.%Y.'));

INSERT INTO stavka_racun VALUES (41, 31, 21, 2),
                                (42, 31, 22, 5),
                                (43, 32, 22, 1),
                                (44, 32, 23, 1);
```

Relax dataset:

```
group: trgovina

kupac = {
  id, ime, prezime
  1, 'Lea', 'Fabris'
  2, 'David', 'Sirotić'
  3, 'Tea', 'Bibić'
}

zaposlenik = {
  id, ime, prezime, oib, datum_zaposlenja
  11, 'Marko', 'Marić', '123451', '01.10.2020.'
  12, 'Toni', 'Milovan', '123452', '02.10.2020.'
  13, 'Tea', 'Marić', '123453', '02.10.2020.'
}

artikl = {
  id, naziv, cijena
  21, 'Puding', 5.99
  22, 'Milka čokolada', 30.00
  23, 'Čips', 9
}

racun = {
  id, id_zaposlenik, id_kupac, broj, datum_izdavanja
  31, 11, 1, '00001', '05.10.2020.'
  32, 12, 2, '00002', '06.10.2020.'
  33, 12, 1, '00003', '06.10.2020.'
}

stavka_racun = {
  id, id_racun, id_artikl, kolicina
  41, 31, 21, 2
  42, 31, 22, 5
  43, 32, 22, 1
  44, 32, 23, 1
}
```

Ugrubo prioriteti izvođenja operacija:

1. selekcija
2. projekcija
3. join
4. agregacija
5. unija/presjek

Osnovna pravila ekvivalencije:

Pravilo	Opis
$\sigma_{\theta_1 \wedge \theta_2}(E) = \sigma_{\theta_1}(\sigma_{\theta_2}(E))$	konjunktivna operacija selekcije može biti rastavljena u niz pojedinačnih selekcija
$\sigma_{\theta_2}(\sigma_{\theta_1}(E)) = \sigma_{\theta_1}(\sigma_{\theta_2}(E))$	operacija selekcije je komutativna
$\Pi_{L_1}(\Pi_{L_2}(\ldots \Pi_{L_n}(E))) = \Pi_{L_1}(E)$	samo je posljednja operacija projekcije u nizu dovoljna, prethodne nisu potrebne
$\sigma_\theta(E_1 \times E_2) = E_1 \bowtie_\theta E_2$ $\sigma_{\theta_1}(E_1 \bowtie_{\theta_2} E_2) = E_1 \bowtie_{\theta_1 \wedge \theta_2} E_2$	selekcije se mogu kombinirati s Kartezijevim produktom i theta join-om
$E_1 \bowtie_\theta E_2 = E_2 \bowtie_\theta E_1$	theta i natural join operacije su komutativne
$(E_1 \bowtie E_2) \bowtie E_3 = E_1 \bowtie (E_2 \bowtie E_3)$	natural join operacije su asocijativne
$(E_1 \bowtie_{\theta_1} E_2) \bowtie_{\theta_2 \wedge \theta_3} E_3 = E_1 \bowtie_{\theta_1 \wedge \theta_3} (E_2 \bowtie_{\theta_2} E_3)$	theta join operacije pod sljedećim uvjetom (*gdje θ_2 uključuje atribute samo iz E_2 i E_3*)
Operacija selekcije se distribuira preko theta join operacije	**pod uvjetima:**
$\sigma_{\theta_0}(E_1 \bowtie_\theta E_2) = (\sigma_{\theta_0}(E_1)) \bowtie_\theta E_2$	svi atributi unutar θ_0 se odnose samo na atribute iz jednog izraza koji se spaja
$\sigma_{\theta_1 \wedge \theta_2}(E_1 \bowtie_\theta E_2) = (\sigma_{\theta_1}(E_1)) \bowtie_\theta (\sigma_{\theta_2}(E_2))$	svi atributi unutar θ_1 se odnose samo na atribute iz E_1, a svi atributi unutar θ_2 se odnose samo na atribute iz E_2
Operacija projekcije se distribuira preko theta join operacije	**pod uvjetima:**
$\Pi_{L_1 \cup L_2}(E_1 \bowtie_\theta E_2) = (\Pi_{L_1}(E_1)) \bowtie_\theta (\Pi_{L_2}(E_2))$	ako θ uključuje samo atribute iz L_1 unija L_2
$\Pi_{L_1 \cup L_2}(E_1 \bowtie_\theta E_2) = \Pi_{L_1 \cup L_2}((\Pi_{L_1 \cup L_3}(E_1))) \bowtie_\theta (\Pi_{L_2 \cup L_4}(E_2))$	- uzmimo u obzir join $E_1 \bowtie_\theta E_2$ - neka su L_1 i L_2 skup atributa iz E_1 i E_2 respektivno - neka su L_3 atributi iz E_1 i L_4 atributi iz E_2 koji sudjeluju u join uvjetu L_4 atributi iz E_2 θ ali ne sudjeluju u L_1 unija L_2 i
$E_1 \cup E_2 = E_2 \cup E_1$ $E_1 \cap E_2 = E_2 \cap E_1$	operacije unije i presjeka su komutativne
$(E_1 \cup E_2) \cup E_3 = E_1 \cup (E_2 \cup E_3)$ $(E_1 \cap E_2) \cap E_3 = E_1 \cap (E_2 \cap E_3)$	operacije unije i presjeka su asocijativne
$\sigma_\theta(E_1 - E_2) = \sigma_\theta(E_1) - \sigma_\theta(E_2)$ $\sigma_\theta(E_1 - E_2) = \sigma_\theta(E_1) - E_2$	operacije selekcije se distribuira preko operacije unije, presjeka i razlike
$\Pi_L(E_1 \cup E_2) = (\Pi_l(E_1)) \cup (\Pi_L(E_2))$	operacije projekcije se distribuira preko operacije unije

Zadatak: Napiši optimizirani plan izvođenja upita koji će prikazati *ime* i *prezime* kupaca čije je ime *'Tea'*:

```
SELECT ime, prezime
  FROM kupac
  WHERE ime = 'Tea';
```

> Rješenje:
> $\sigma_{ime='Tea'}(\Pi_{ime,\ prezime}(kupac))$
>
> ✅ Optimizirano rješenje:
> $\Pi_{ime,\ prezime}(\sigma_{ime='Tea'}(kupac))$

Zadatak: Napiši optimizirani plan izvođenja upita koji će prikazati *ime* i *prezime* kupca čije je ime *'Tea'* i prezime *'Sirotić'*

```
SELECT ime, prezime
  FROM kupac
  WHERE ime = 'Tea' AND prezime = 'Sirotić';
```

> Rješenje:
>
> $\sigma_{ime='Tea' \wedge prezime='Sirotić'}(\Pi_{ime,\ prezime}(kupac))$
>
> ✅ Optimizirano rješenje:
>
> $\Pi_{ime,\ prezime}(\sigma_{ime='Tea' \wedge prezime='Sirotić'}(kupac))$

Zadatak: Napiši optimizirani plan izvođenja upita koji će prikazati *ime* i *prezime* kupca čije je ime *'Tea'* ili prezime *'Sirotić'*

```
SELECT ime, prezime
  FROM kupac
  WHERE ime = 'Tea' OR prezime = 'Sirotić';
```

> ❌ Pogrešno rješenje:
>
> $\Pi_{ime,\ prezime}(\sigma_{ime='Tea'}(\sigma_{prezime='Sirotić'}(kupac)))$
>
> ✅ Optimizirano rješenje:
>
> $\Pi_{ime,\ prezime}(\sigma_{ime='Tea' \vee prezime='Sirotić'}(kupac))$

Zadatak: Napiši optimizirani plan izvođenja upita koji će prikazati *naziv* artikla i *količinu* u kojoj je on izdan na stavkama računa gdje je *naziv* artikla *'Milka čokolada'* i *količina* veća od *2*.

```
SELECT naziv, kolicina
  FROM artikl AS a
  INNER JOIN stavka_racun AS sr ON sr.id_artikl = a.id
  WHERE naziv = 'Milka čokolada' AND kolicina > 2;
```

> Rješenje:
> $\sigma_{naziv='Milka\ čokolada' \wedge kolicina>2}(\Pi_{naziv,\ kolicina}(stavka\_racun \bowtie_{artikl.id=id\_artikl} artikl))$
>
> ✅ Optimizirano rješenje:
> $\Pi_{naziv,\ kolicina}((\sigma_{naziv='Milka\ čokolada'}(artikl)) \bowtie_{artikl.id=id\_artikl} \Pi_{id\_artikl,\ kolicina}(\sigma_{kolicina>2}(stavka\_racun)))$

Zadatak: Napiši optimizirani plan izvođenja upita koji će prikazati najveću *cijenu* artikala koji imaju cijenu manju od *10*.

```
SELECT MAX(cijena) AS cijena
  FROM artikl
  WHERE cijena < 10;
```

> ❌ Pogrešno rješenje: $\sigma_{cijena<10}(\gamma_{;max(cijena)->cijena}(artikl))$
>
> ✅ Optimizirano rješenje: $\gamma_{;max(cijena)->cijena}(\sigma_{cijena<10}(artikl))$

Zadatak: Napiši optimizirani plan izvođenja upita koji će prikazati *imena* i *prezimena* kupaca i zaposlenika koji imaju jednako *ime*.

```
SELECT k.ime, k.prezime, z.ime, z.prezime
  FROM kupac AS k, zaposlenik AS z
  WHERE k.ime = z.ime;
```

> Rješenje:
> $\sigma_{kupac.ime=zaposlenik.ime}(\Pi_{kupac.ime,\ kupac.prezime,\ zaposlenik.ime,\ zaposlenik.prezime}(kupac \times zaposlenik))$
>
> ✅ Optimizirano rješenje:
>
> $\Pi_{ime,\ prezime}(kupac) \bowtie_{kupac.ime=zaposlenik.ime} \Pi_{ime,\ prezime}(zaposlenik)$

Zadatak: Napiši optimizirani plan izvođenja upita koji će prikazati *brojeve* računa i ukupnu *količinu* artikla *'Milka čokolada'* izdanu na pojedinom računu, a pritom prikazati samo račune koji imaju *broj* manji od *'00003'*.

```
SELECT broj, SUM(kolicina) AS kolicina_cokolade
  FROM racun AS r
  INNER JOIN stavka_racun AS sr ON sr.id_racun = r.id
  INNER JOIN artikl AS a ON sr.id_artikl = a.id
  WHERE broj < '00003' AND naziv = 'Milka čokolada'
  GROUP BY r.id;
```

> Rješenje:
> $x = \sigma_{broj<'00003' \wedge naziv='Milka\ čokolada'}(racun \bowtie_{racun.id=id\_racun} stavka\_racun \bowtie_{id\_artikl=artikl.id} artikl)$
>
> $\Pi_{broj,\ kolicina\_cokolade}(\gamma_{racun.id,\ broj;\ sum(kolicina)->kolicina\_cokolade}(x))$
>
> ✅ Optimizirano rješenje:
>
> $rac = \sigma_{broj<'00003'}(racun)$
>
> $art = \sigma_{naziv='Milka\ čokolada'}(artikl)$
>
> $x = rac \bowtie_{racun.id=id\_racun} stavka\_racun \bowtie_{id\_artikl=artikl.id} art$
>
> $\Pi_{broj,\ kolicina\_cokolade}(\gamma_{racun.id,\ broj;\ sum(kolicina)->kolicina\_cokolade}(x))$

Zadatak: Napiši optimizirani plan izvođenja upita koji će prikazati popis *imena* i *prezimena* kupaca i zaposlenika čije je *ime 'Tea'* (rezultat su dva stupca: *ime*, *prezime*).

```
SELECT ime, prezime
  FROM kupac
  WHERE ime = 'Tea'
UNION
SELECT ime, prezime
  FROM zaposlenik
  WHERE ime = 'Tea';
```

> Rješenje: $\sigma_{ime='Tea'}(\Pi_{ime,\,prezime}(kupac) \cup \Pi_{ime,\,prezime}(zaposlenik))$
>
> ✅ Optimizirano rješenje: $(\Pi_{ime,\,prezime}(\sigma_{ime='Tea'}(kupac))) \cup (\Pi_{ime,\,prezime}(\sigma_{ime='Tea'}(zaposlenik)))$

Dekorelacija:

```
SELECT *
  FROM zaposlenik
  WHERE EXISTS (SELECT id_zaposlenik FROM racun WHERE zaposlenik.id = id_zaposlenik);
```

> Rješenje:
>
> ```
> SELECT DISTINCT z.*
>   FROM zaposlenik AS z
> LEFT JOIN racun AS r ON r.id_zaposlenik = z.id
> WHERE r.id_zaposlenik IS NOT NULL;
> ```
>
> ✅ Optimizirano rješenje:
>
> $\Pi_{zaposlenik.id,\,ime,\,prezime,\,oib,\,datum\_zaposlenja}(\sigma_{id\_zaposlenik!=null}(zaposlenik \bowtie_{id\_zaposlenik=zaposlenik.id} racun))$

Optimizacija komplicirano riješenog zadatka: Napiši proceduru koja će u izlaznu varijablu spremiti vrijednost *'DA'* ako svi zaposlenici imaju točno **11** znakova u oib-u, dok će u suprotnom (barem jedan korisnik nema oib dužine **11** znakova) u izlaznu varijablu spremiti vrijednost *'NE'*.

> Rješenje:

```
DELIMITER //
CREATE PROCEDURE provjeri_oib(OUT rezultat VARCHAR(2))
BEGIN
  DECLARE oib varchar(20);
  DECLARE finished INT DEFAULT 0;

  DECLARE cur CURSOR FOR
    SELECT oib FROM zaposlenik;

  DECLARE EXIT HANDLER FOR NOT FOUND SET finished = 1;

  OPEN CUR;

  iteriraj: LOOP

    FETCH cur INTO oib;

    IF finished = 1 THEN
      LEAVE iteriraj;
    END IF;

    IF LENGTH(oib) < 11 THEN
      SET rezultat ='NE';
      LEAVE iteriraj;
    END IF;

  END LOOP iteriraj;
  CLOSE cur;
END //
DELIMITER ;

CALL provjeri_oib(@rezultat);
SELECT @rezultat FROM DUAL;
```

> ✅ Optimizirano rješenje:

```
DELIMITER //
CREATE PROCEDURE provjeri_oib(OUT rezultat VARCHAR(2))
BEGIN
  DECLARE broj_neispravnih INTEGER;

  SELECT COUNT(*) INTO broj_neispravnih
    FROM zaposlenik
    WHERE LENGTH(oib) < 11;

  IF broj_neispravnih = 0 THEN
    SET rezultat ='DA';
  ELSE
    SET rezultat ='NE';
  END IF;

END //
DELIMITER ;

CALL provjeri_oib(@rezultat);
SELECT @rezultat FROM DUAL;
```

Zadaci:

1. Napiši upit i optimizirani plan izvođenja upita koji će prikazati *nazive* i *cijene* artikala čija je cijena u rasponu *5* i *20* (tj. **[5, 20]**).
2. Napiši upit i optimizirani plan izvođenja upita koji će prikazati *prezimena* svih zaposlenika zajedno sa *nazivima* artikala koje su izdali kroz račune, a pritom pritom prikazati samo zaposlenike sa prezimenom *'Marić'* i artikle sa cijenom većom od *20*.
3. Napiši optimizirani plan izvođenja sljedećeg izraza:
   $\sigma_{(ime='Tea' \vee prezime='Sirotić') \wedge broj>'00001'}(\Pi_{ime,\, prezime,\, broj}(kupac \bowtie_{kupac.id=id\_kupac} racun))$