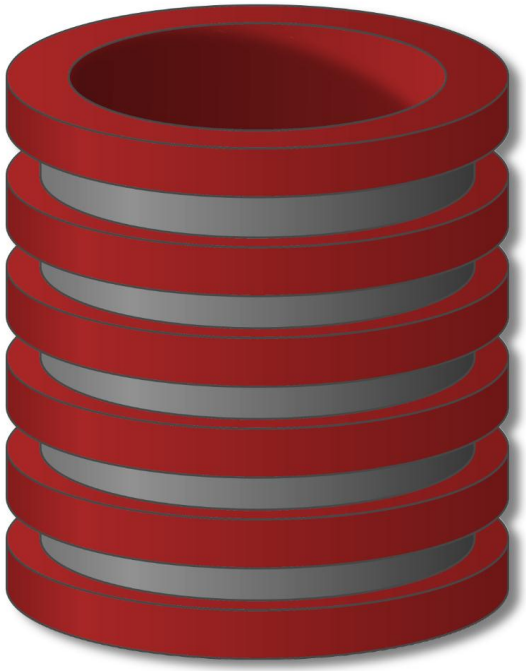


MySQL

Procedure deo 2

MySQL Parametri Store Procedure

- Skoro uvek će store procedure koje razvijamo zahtevati parametre.
- Parametri obezbeđuju store proceduri fleksibilnost i korisnost.
- U MySql-u parametar može da ima jednu od tri forme:
  - **IN**
  - **OUT**
  - **INOUT**

MySQL IN Parametar u Store Proceduri

```
DELIMITER //
CREATE PROCEDURE FilijalePoZemljama (IN imeZemlje VARCHAR(255))
BEGIN
        SELECT *
        FROM filijale
        WHERE zemlja = imeZemlje;
END //
DELIMITER ;
```

```
CALL FilijalePoZemljama (‘Srbija');
```

```
CALL FilijalePoZemljama (‘Francuska');
```

MySQL OUT Parametar u Store Proceduri

```
DELIMITER $$
CREATE PROCEDURE BrojPordzbinaStatus( IN StatusPordzb VARCHAR(25), OUT ukupno INT)
BEGIN
        SELECT count(*) INTO ukupno
        FROM Porudzbine
        WHERE status = StatusPordzb;
END$$
DELIMITER ;
```

```
CALL BrojPordzinaStatus('Shipped',@ukupno);
SELECT @ukupno;
```

```
CALL BrojPordzinaStatus('in process',@ukupno);
SELECT @ukupno AS  'ukupno u izradi' ;
```

MySQL INOUT Parametar u Store Proceduri

```
DELIMITER $$
CREATE PROCEDURE set_brojac(INOUT brojac INT(4),IN inc INT(4))
BEGIN
SET brojac = brojac + inc;
END$$
DELIMITER ;
```

```
SET @brojac = 1;
CALL set_brojac(@brojac,1);   -- 2
CALL set_brojac(@brojac,1);   -- 3
CALL set_brojac(@brojac,5);   -- 8
SELECT @brojac;   -- 8
```

MySQL Store Procedure koje vraćaju više parametara

- MySQL stored procedure vraćaju samo jednu vrednost.
- Da bi se kreirala procedura koja vraća više vrednosti, potrebno je da se koristi procedura sa INOUT ili OUT parametrima.

```
DELIMITER $$
CREATE PROCEDURE get_order_by_cust(
IN cust_no INT,
OUT shipped INT,
OUT canceled INT,
OUT resolved INT,
OUT disputed INT)
BEGIN
-- shipped
SELECT
        count(*) INTO shipped
FROM
        orders
WHERE
        customerNumber = cust_no
        AND status = 'Shipped';
```

Ulazna promenjiva preko koje saopštavamo upitu vrednost.

Izlazne promenjive u kojima čuvamo rezlutate **select** upita

Rezlutat upita čuvamo u promenjivu shipped

Upitu saopštavamo ulazni parametar

```
-- canceled
SELECT
        count(*) INTO canceled
 FROM
        orders
 WHERE
        customerNumber = cust_no
         AND status = 'Canceled';
 -- resolved
SELECT
        count(*) INTO resolved
 FROM
        orders
 WHERE
        customerNumber = cust_no
         AND status = 'Resolved';
```

MySQL Store Procedure koje vraćaju više parametara

```
-- disputed
SELECT
        count(*) INTO disputed
FROM
        orders
WHERE
        customerNumber = cust_no
           AND status = 'Disputed';
END
```

```
CALL get_order_by_cust(141,@shipped,@canceled,@resolved,@disputed);

SELECT @shipped,@canceled,@resolved,@disputed;
```

Šema Baze podataka

Browse | Structure | SQL | Search | Insert | Export | Import | Privileges | Operations | Triggers

#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
1	IdRadnik	int(11)			No	None	AUTO_INCREMENT	Change Drop Primary Unique Index Spatial Fulltext Distinct values
2	Ime	varchar(50)	latin1_swedish_ci		No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values
3	Pol	char(1)	latin1_swedish_ci		No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values
4	Plata	int(11)			Yes	NULL		Change Drop Primary Unique Index Spatial Fulltext Distinct values
5	Grad	varchar(30)	latin1_swedish_ci		Yes	NULL		Change Drop Primary Unique Index Spatial Fulltext Distinct values
6	Email	varchar(30)	latin1_swedish_ci		Yes	NULL		Change Drop Primary Unique Index Spatial Fulltext Distinct values
7	IdOdeljenje	int(11)			Yes	NULL		Change Drop Primary Unique Index Spatial Fulltext Distinct values

	IdRadnik	Ime	Pol	Plata	Grad	Email	IdOdeljenje
Edit Copy Delete	1	Milorad	m	55000	Nis	milorad_mancic@yahoo.com	1
Edit Copy Delete	2	Anastasija	z	60000	Nis	ana@yahoo.com	1
Edit Copy Delete	3	Natasa	z	45000	Beograd	nata@gmail.com	2
Edit Copy Delete	4	Aca	m	34600	Aleksinac	aca@hotmail.com	3
Edit Copy Delete	6	Bosko	m	75000	Beograd	boki@gmail.com	4
Edit Copy Delete	7	Nikola	m	100000	Novi sad	NULL	NULL
Edit Copy Delete	8	Jovana	z	39000	Nis	NULL	NULL

MySQL Store Procedure Ulazni Parametri Primeri

SQL File 4* | new_procedure - Routine | spPrikazRadnikaQ | new_procedure - Routine

Limit to 1000 rows

```
1  DELIMITER $$
2  CREATE PROCEDURE spPrikazRadnikaIN(in INpol char(1), in INgrad varchar(30))
3
4  BEGIN
5  select *
6  from tblradnik
7  where pol = INpol and grad=INgrad;
8  END
```

SQL File 4* | spPrikazRadnikaQ

Limit to 1000 rows

```
1  call dbevidencija.spPrikazRadnikaIN('m','nis')
2
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

idRadnik	Ime	Pol	Plata	Grad	IdOdeljenje	Email	Sef
1	Nikola	m	55000	Nis	1	nikola@gmail.com	2
2	Stefan	m	35000	Nis	NULL	stefan@hotmail.com	NULL
7	Miroslav	m	70000	Nis	3	miroslav@gmail.com	6

SQL File 4* | spPrikazRadnikaQ

Limit to 1000 rows

Find

```
1  set @pol = 'm';
2  set @grad = 'Nis';
3  call dbevidencija.spPrikazRadnikaIN(@pol,@grad)
4
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

idRadnik	Ime	Pol	Plata	Grad	IdOdeljenje	Email	Sef
1	Nikola	m	55000	Nis	1	nikola@gmail.com	2
2	Stefan	m	35000	Nis	NULL	stefan@hotmail.com	NULL
7	Miroslav	m	70000	Nis	3	miroslav@gmail.com	6

MySQL Store Procedure Ulazni/ Izlazni Parametri Primeri

Run SQL query/queries on database evidencija:

```
1  DELIMITER //
2  CREATE PROCEDURE spInOutBrojOSoba(IN sex char(1),OUT BrojOsoba int)
3  BEGIN
4      SELECT count(*) INTO BrojOsoba
5       FROM `tblradnik`
6      WHERE Pol = sex;
7   END //
8  DELIMITER ;
```

```
SET @p0='z'; CALL `spInOutBrojOSoba`(@p0, @p1); SELECT @p1 AS `BrojOsoba`;
```

Execution results of routine `spInOutBrojOSoba`

BrojOsoba
3

```
SET @p0='m'; CALL `spInOutBrojOSoba`(@p0, @p1); SELECT @p1 AS `BrojOsoba`;
```

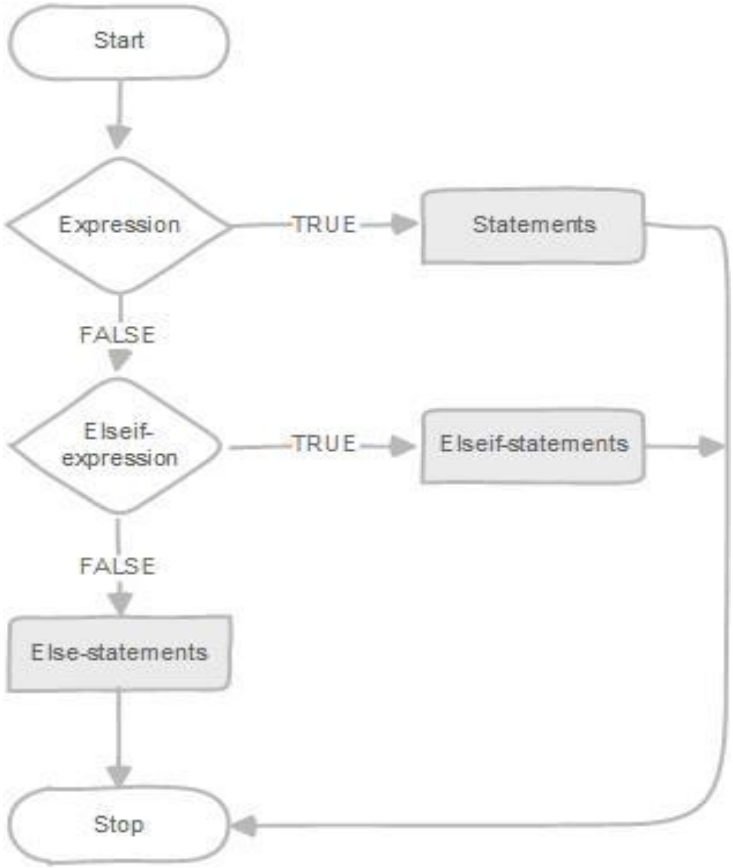
Execution results of routine `spInOutBrojOSoba`

BrojOsoba
5

MySQL **IF** ISKAZ

- MySQL IF iskaz dozvoljava da se izvrši skup SQL iskaza na osnovu toga da li je uslov ispunjen
- IF Iskaz može da vrati jednu od tri vrednosti TRUE FALSE ili NULL.

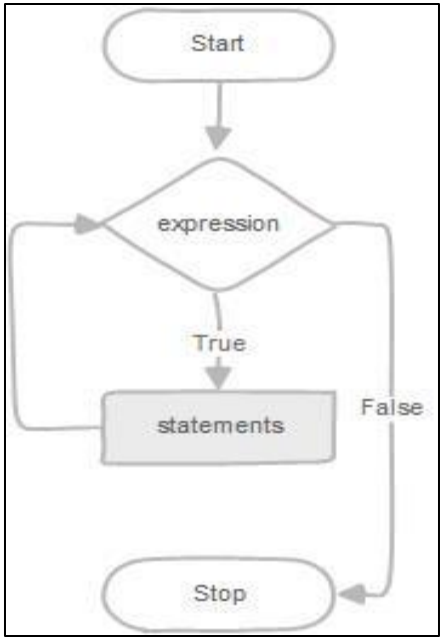
```
IF uslov THEN
  iskazi;
ELSEIF elseif-uslov THEN
  elseif-iskazi;
...
ELSE
  else-iskazi;
END IF;
```



MySQL WHILE LOOP

- MySQL obezbeđuje izvršavanje iskaza (SQL koda) u petlji sve dok je uslov ispunjen
- Postoje tri loop tehnike:
  - **WHILE**
  - **REPEAT**
  - **LOOP**

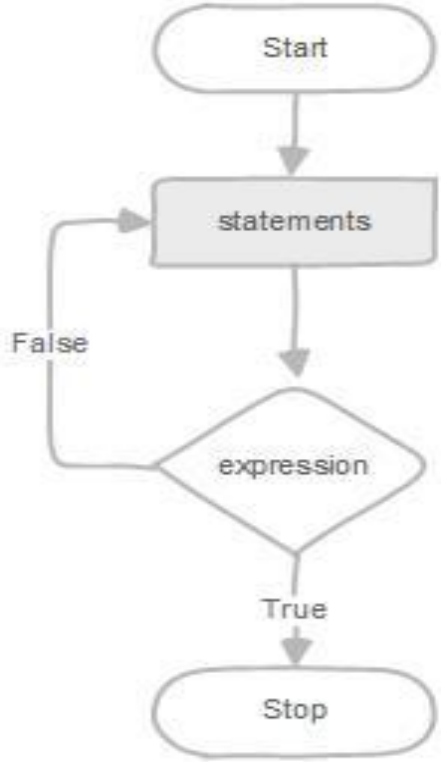
```
WHILE uslov DO
   iskazi;
END WHILE;
```



MySQL REPEAT LOOP

- U ovom slučaju, MySQL prvo izvršava iskaze a zatim ispituje uslov.
- MySQL prolazi kroz petlju sve dok uslov nije ispunjen tj. sve dok vraća FALSE

```
REPEAT
iskaz;
UNTIL uslov
END REPEAT;
```



ITERATE i LEAVE Iskazi

- Koriste se dva iskaza za kontrolu loop petlje
  - **LEAVE** iskaz koji odamah napušta petlju, slično **BREAK** iskazu u višim programskim jezicima
  - **ITERATE** iskaz preskače ceo kod ispod i prelazi na novu iteraciju, slično **CONTINUE** iskazu u višem programskom jeziku.

LOOP Petlja

- U MySQL-u postoji i LOOP iskaz koji izvršava blok komandi koristeći loop labelu.

```
CREATE PROCEDURE test_loop()
BEGIN
  DECLARE x  INT;
  DECLARE str  VARCHAR(255);

  SET x = 1;
  SET str =  '';

  loop_label:  LOOP
  IF  x > 10 THEN
    LEAVE  loop_label;
  END  IF;
```

```
SET  x = x + 1;
IF  (x mod 2) THEN
ITERATE  loop_label;
ELSE
            SET  str = CONCAT(str,x,',');
END  IF;
       END LOOP;

       SELECT str;

END;
```

CASE ISKAZ

- Pregleda listu uslova i vraća jednu od više ponuđenih vrednosti

Prost CASE iskaz:

```
CASE ulazna vrednost (Ime Kolone)
WHEN vrednost THEN rezlutat[ ...n ]
[ELSE vrednost]
END
```

- Ulazna vrednost se poredi sa vrednošću u When iskazu.
- Čim se naiđe na prvi tačan iskaz izvršava se rezlutat u Then iskazu za taj red

Search CASE iskaz:

```
CASE
WHEN Boolean_expression THEN result_expression [ ...n ] ELSE
else_result_expression ]
END
```

- Svaki When iskaz vraća boolen vrednost.
- Ukoliko nema true iskaza isvršava se vrednost u else iskazu ukoliko postoji.
- Ukoliko nema true iskaza i elese iskaza rezlutat je null vrednost

CASE ISKAZ

ZADATAK 1

Prikazati za svaki grad u kome radnik radi kolonu sa skaraćenim nazivom grada

Id	Ime	Pol	Plata	Grad	OdelenjeId	sef
1	Marko	Muskarac	45000	Nis	1	3
2	Ana	Zena	38000	Beograd	2	5
3	Darko	Muskarac	67000	Nis	1	NULL
4	Sanja	Zena	56000	Novi Sad	2	NULL
5	Danijela	Zena	69000	Beograd	4	NULL
6	Mirko	Muskarac	100000	Novi Sad	3	4
7	Zlatko	Muskarac	56000	Novi Sad	NULL	4
8	Milica	Zena	76000	Nis	NULL	3

Ime	Grad	Skraceni naziv
Marko	Nis	Ni
Ana	Beograd	Bg
Darko	Nis	Ni
Sanja	Novi Sad	Ns
Danijela	Beograd	Bg
Mirko	Novi Sad	Ns
Zlatko	Novi Sad	Ns
Milica	Nis	Ni

```
Select R.Ime, R.Grad,
Case r.grad
When 'Nis' Then 'Ni'
When 'Beograd' Then 'Bg'
When 'Novi Sad' Then 'Ns'
Else 'Nepoznata Skracenica'
End
as [Skraceni naziv]
From tblRadnik R
```

CASE ISKAZ

ZADATAK 2:

Prikazati za svakog menadzera u kom gradu je menadzer

Id	Ime	Pol	Plata	Grad	OdelenjeId	sef
1	Marko	Muskarac	45000	Nis	1	3
2	Ana	Zena	38000	Beograd	2	5
3	Darko	Muskarac	67000	Nis	1	NULL
4	Sanja	Zena	56000	Novi Sad	2	NULL
5	Danijela	Zena	69000	Beograd	4	NULL
6	Mirko	Muskarac	100000	Novi Sad	3	4
7	Zlatko	Muskarac	56000	Novi Sad	NULL	4
8	Milica	Zena	76000	Nis	NULL	3

Ime Radnika	Menadzer
Marko	Darko
Ana	Danijela
Darko	Menadzer iz Nisa
Sanja	Menadzer iz Novog Sada
Danijela	Menadzer iz Beograda
Mirko	Sanja
Zlatko	Sanja
Milica	Darko

```
Select R.ime as [Ime Radnika],
Case
When (M.Ime is null and R.Grad='Nis') Then 'Menadzer iz Nisa'
When (M.ime is null and R.Grad='Beograd') Then 'Menadzer iz Beograda'
When (M.ime is null and R.Grad='Novi Sad') Then 'Menadzer iz Novog Sada'
Else M.ime
End as Menadzer
From tblRadnik r
Left join tblRadnik M
On R.sef=M.Id
```

PRIMER: Kolona koja uvek sadrži vrednosti od 1 do ukupnog broja redova a nije auto_increment – aktivira se prilikom brisanja a može i prilikom dodavanja novog reda

```
DELIMITER $$
CRETAE PROCEDURE Azuriraj()
BEGIN
  DECLARE ispitaj_id int(4);
  DECLARE inkrement int(4);
  DECLARE maxid int(4);

  SET ispitaj_id=1;
  SET inkrement=1;
  SET maxid=(SELECT max(id) FROM NIVOI);

  WHILE ispitaj_id<=maxid DO
     IF ispitaj_id=(SELECT id FROM NIVOI WHERE id=ispitaj_id) THEN
              UPDATE NIVOI SET idn=inkrement WHERE id=ispitaj_id;
               SET inkrement=inkrement+1;
      END IF;
      SET ispitaj_id=ispitaj_id+1;
  END WHILE;
END $$
DELIMITER ;
```

PRIMER: Kolona koja uvek sadrži vrednosti od 1 do ukupnog broja redova a nije auto_increment – aktivira se prilikom dodavanja novog reda

```
DELIMITER $$
CRETAE PROCEDURE Azuriraj()
BEGIN
  DECLARE maxid int(4);
  DECLARE maxidn int(4);

  SET maxid = ( SELECT max(id) FROM nivo);
  SET maxidn = (SELECT max(idn)+1 FROM nivo);

  UPDATE nivo SET idn=maxidn WHERE id=maxid;


END $$
DELIMITER ;
```

My SQL CASE ISKAZI

- MySQL CASE iskazi se koriste za kreiranje kompleksnih uslovnih iskaza.
- Alternativa IF iskazima su CASE iskazi.
- CASE iskazi obezbeđuju bolju čitljivost i veću efikasnost koda.
- Koriste se dve varijante CASE iskaza:
  - **SIMPLE CASE**
  - **SEARCHED CASE**

.

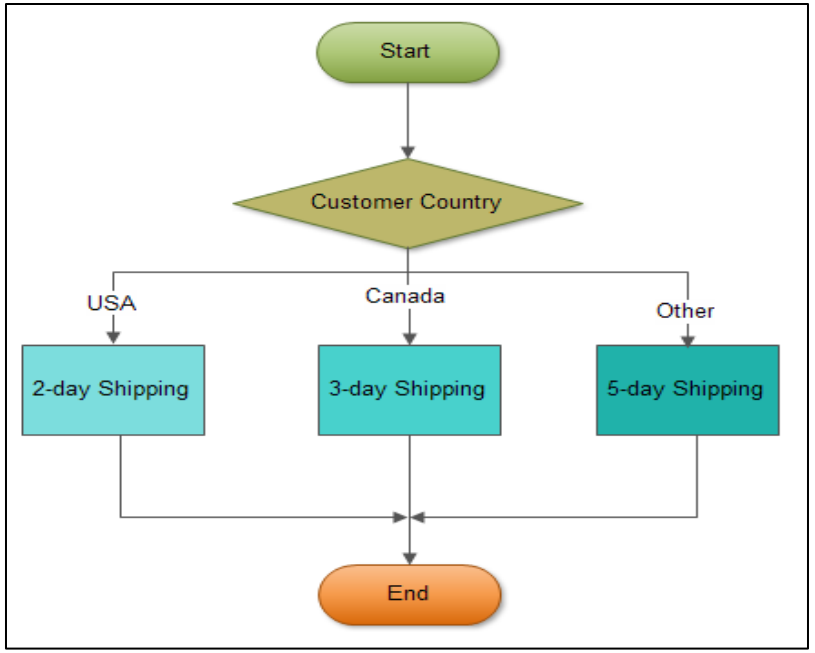
SIMPLE CASE

```
CASE  case_expression
  WHEN when_expression_1 THEN commands
  WHEN when_expression_2 THEN commands
  ...
  ELSE commands
END CASE;
```

Primer SIMPLE CASE ISKAZ

```
DELIMITER $$
 CREATE PROCEDURE GetCustomerShipping(
in  p_customerNumber int(11),
out p_shiping       varchar(50))
BEGIN
DECLARE customerCountry varchar(50);
 SELECT country INTO customerCountry
FROM customers
WHERE customerNumber = p_customerNumber;

CASE customerCountry
   WHEN  'USA' THEN
      SET p_shiping = '2-day Shipping';
   WHEN 'Canada' THEN
      SET p_shiping = '3-day Shipping';
    ELSE
       SET p_shiping = '5-day Shipping';
END CASE;
END$$
```



- ELSE klauzula je opciona.
- Ukoliko se ELESE klauzula izostavi i ne pronađe se ni jedno mečovanje My SQL prijaviće grešku.

```
SET @customerNo = 112;

SELECT country into @country
FROM customers
WHERE customernumber = @customerNo;

CALL GetCustomerShipping(@customerNo,@shipping);

SELECT @customerNo AS Customer,
      @country    AS Country,
      @shipping   AS Shipping;
```

Primer SEARCHED CASE ISKAZA

```
CASE
   WHEN condition_1 THEN commands
   WHEN condition_2 THEN commands
   ...
   ELSE commands
END CASE;
```

- Simple Case nam dozvoljava da mečujemo iskaz ispitujući različite vrednosti
- Da bismo izvršavali kompleksna mečovanja koristi se Searched Case iskazi.
- Sličan je IF iskazu samo što je čitljiviji.

IZBOR IZMEĐU IF i CASE ISKAZA

- MySQL obezbeđuje oba IF i CASE iskaze za izvršenje skupa SQL iskaza na osnovu ispunjenog uslova.
- Za većinu developera razlika je samo u ličnnim preferencijalima
- Prilikom uptrebe potrebno je razmotriti sledeće:
  - Simple CASE iskaz je čitljiviji u odnosu na IF iskaz kada se upoređuje samo jedan iskaz sa opsegom vrednosti.
  - Takođe je i efikasniji.
  - Kada se proveravaju kompleksni iskazi na osnovu više vrednosti, IF iskaz je jednostavnije razumeti.

MySQL CURSOR

- Da bi upravljali skupom rezlutata (result set) unutar store procedure koristimo kursor.
- Kursor nam omogućava da prolazimo kroz redove koje vraća upit i procesira svaki red
- MySQL kursor je:
  - **Read only**: ne možemo da radimo update podataka u tabeli kroz kursor.
  - **Non-scrollable**: možemo da obrađujemo (fetch) redove u redosledu koji je definisan SELECT iskazom, u obrnutom redosledu ne možemo.
    - Ne možemo da preskočimo redove da bi stigli do željenog reda u result set-u.
  - **Asensitive**: postoje dve vrste kursora: asensitive cursor i insensitive cursor.
    - Asensitive cursor ukazuje na stvarni podatak, dok insensitive cursor koristi privremenu kopiju podatka.
    - Asensitive cursor izvršava se brže od insensitive cursor jer ne mora da pravi privremenu kopiju podatka.
    - Međutim, bilo koja promena koja je načinjena nad podatkom od drugih konekcija uticaće na podatak koji se koristi od strane asensitive kursora, bezbednije je ukoliko se podatak ne ažurira.

MySQL CURSOR
I KORAK

- Prvo, potrebno je deklarisati kursor koristeći DECLARE iskaz.

```
DECLARE cursor_name CURSOR FOR SELECT_statement;
```

- Kursor se definiše posle definisanja bilo kojih promenjivih.
- Ukoliko se kursor definiše pre promenjivih MySQL prijaviće grešku.
- Kursor mora uvek da bude povezan sa SELECT iskazom.

MySQL CURSOR
II KORAK

- Nakon definisanja kursora potrebno je kursor otvoriti naredbom OPEN.
- OPEN iskaz vrši inicijalizaciju result set za kursor.
- Da bi se koristili redovi iz result seta potrebno je prvo otvoriti kursor.

```
OPEN cursor_name;
```

MySQL CURSOR
III KORAK

- FETCH iskaz se koristiti da bi se izvadio sledeći red od strane kursora i da bi se kursor pomerio na sledeći red u result set-u

```
FETCH cursor_name INTO variables list;
```

- Potrebno je proveriti da li postoji dostupan red koji se može izvaditi(fetch)

MySQL CURSOR
IV KORAK

- Na kraju se poziva CLOSE iskaz kako bi se kursor deaktivirao i oslobodio memoriju koju je koristio.

```
CLOSE cursor_name;
```

- Prilikom rada sa MySQL kursorom potrebno je definisati NOT FOUND *handler* da bi smo rešili situaciju kada kursor ne može da pronađe ni jedan red.
- Svaki put kada se pozove FETCH iskaz, kursor pokušava da pročita sledeći red u *result set*.
- Kada kursor stigne do kraja u result setu, on ne može da pribavi podatak, handler se koristi da obradi uslov koji se javlja kada nema više redova u result setu.

MySQL CURSOR
V KORAK

- Prilikom rada sa MySQL kursorom potrebno je definisati NOT FOUND *handler* da bi smo rešili situaciju kada kursor ne može da pronađe ni jedan red.
- Svaki put kada se pozove FETCH iskaz, kursor pokušava da pročita sledeći red u *result set*.
- Kada kursor stigne do kraja u result setu, on ne može da pribavi podatak, handler se koristi da obradi uslov koji se javlja kada nema više redova u result setu.

```
DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;
```

- Finished je promenjiva koja ukazuje da je kursor stigao do kraja result seta.
- Definicija handlera mora da se javi posle definisanja promenjivih i kursora unutra store procedure.

DIJAGRAM RADA MYSQL CURSOR