



Funded by the
Erasmus+ Programme
of the European Union



This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.



KI203 - JAVA 6: NAPREDNO PROGRAMIRANJE U JAVI

Programiranje sa bazama podataka

Lekcija 03

PRIRUČNIK ZA STUDENTE

KI203 - JAVA 6: NAPREDNO PROGRAMIRANJE U JAVI

Lekcija 03

PROGRAMIRANJE SA BAZAMA PODATAKA

- ✓ Programiranje sa bazama podataka
- ✓ Poglavlje 1: Sistemi relacionih baza podataka
- ✓ Poglavlje 2: JDBC
- ✓ Poglavlje 3: PreparedStatement
- ✓ Poglavlje 4: CallableStatement
- ✓ Poglavlje 5: Dobijanje meta podatka o bazi podataka
- ✓ Poglavlje 6: Razvoj Swing GUI aplikacije sa bazom podataka
- ✓ Poglavlje 7: 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

Ciljevi lekcije

Ova lekcija ima za cilj da student nauči da:

1. razume koncept baza podataka i sistema za upravljanje bazama podataka
2. razume relacioni model baze podataka, relacione strukture podataka, ograničenja i jezici,
3. upotrebljavaju SQL za kreiranje i brišu tabele podataka, da nalaze i menjaju podatke,
4. nauče kako da instaliraju i koriste JDBC i da ga koriste za izvršenje naredbi vezanih za korišćenje baza podataka i za obradu rezultata upita poslatih bazama podataka,
5. koriste i izvršavaju prekompilisane SQL iskaze,
6. koriste memorisane SQL procedure i funkcije, i da
7. rade sa baze sa metapodacima upotrebom interfejsa DatabaseMetaData i ResultSetMetaData.

Poglavlje 7, Rayvoj Swing GUI aplikacije sa bazom podataka nije obavezno za studente na CS101 predmetu. Ipak, preporučljivo je da se student upozna sa postupcima razvoja aplikacije koja koristi Swing GUI i MySQL bazu podataka. To poglavlje čine samo video klipovi (na engleskom).

Zbog obima programskog sadržaja i nastavnog materijala, ova lekcija se realizuje u toku dve nedelje (11. i 12. nedelja u semestru)

Referenca: Y. Daniel Liang, INTRODUCTION TO JAVA PROGRAMMING (COMPREHENSIVE VERSION), Tenth Edition, Pearson, ISBN 10: 0-13-376131-2, ISBN 13: 978-0-13-376131-3

Ovo je osnovni udžbenik za ovaj predmet i preporučuje se studentima da ga koriste,

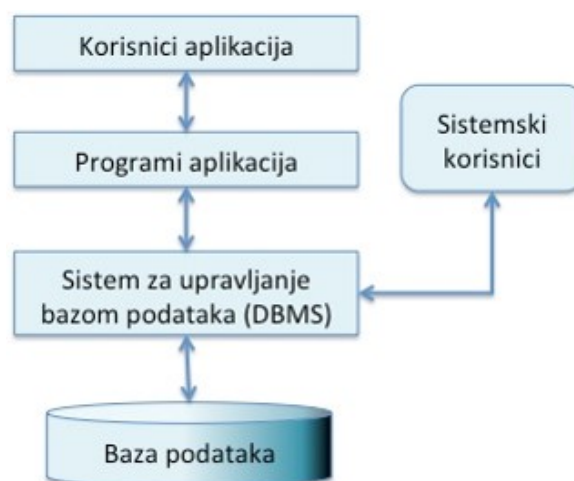
▼ Poglavlje 1

Sistemi relacionih baza podataka

ŠTA SU SISTEMI RELACIONIH BAZA PODATAKA?

Sistem baze podataka čini baza podataka, softver za skladištenje i upravljanje podacima u bazi podataka, aplikacioni programi koji koriste podatke i njihovi krajnji korisnici.

Sistem baze podataka čini baza podataka, softver koji skladištenje i upravljanje podacima u bazi podataka, aplikacioni programi koji koriste podatke i njihovi krajnji korisnici. (slika 1).



Slika 1.1.1 Sistem baze podataka

Baza podataka je skladište podataka koji čine informaciju. **Sistem za upravljanje bazom podataka** (Database Management System - DBMS) je softverski sistem koji koriste programeri aplikacija da bi koristili upravljali podacima u bazama podataka

Aplikacioni programi koriste sisteme za upravljanje bazama podataka, s jedne strane, i rade sa krajnjim korisnicima aplikacija, s druge strane. Aplikacioni programi mogu biti u formi samostalnih GUI aplikacija ili veb aplikacije. One mogu da koriste više sistema baza podataka koji su u mreži računara koje koriste (slika 2).



Slika 1.1.2 Aplikacioni program može koristiti više sistema za upravljanje bazama podataka.

RELACIONE STRUKTURE

Tabele opisuju relacije (odnose) među podacima. Svaki red u tabeli predstavlja jedan slog relacionih podataka. Svaka kolona predstavlja jedan atribut sloga i sadrži njihove vrednosti.

Najveći broj baza podataka danas čine tzv. relacione baze podataka, nazvane po racionom modelu podataka koje primenjuju. Relacioni model podataka čine tri komponente:

- Struktura,
- Integritet (celovitost)
- Jezik

Struktura definiše predstavljanje podataka. Integritet uvodi ograničenja podacima. Jezik obezbeđuje način pristupa i manipulacije sa podacima.

Relacija je jedna tabela koju čine redovi koji ne smeju da se dupliraju /ne smeju da budu isti). Jedan red u tabeli predstavlja jedan slog (record), a kolona tabele sadrži vrednosti podataka, tj. atributa. U teoriji relacionih baza podataka, red u tabeli se naziva torak (tuple), a kolona se naziva atributom. Slika 1 prikazuje jedan primer jedne tabele (relacije) koja skladišti podatke o predmetima univerziteta.

Tabele opisuju relacije (odnose) među podacima. Svaki red u tabeli predstavlja jedan slog relacionih podataka.

Svaki slog može da ima više atributa. Na primer, slog koji sadrži sve podatke o nekom predmetu, može da sadrži njegovu oznaku (npr. KI203), naziv sloga (npr. Objekti i apstrakcija podataka), i broj ECTS kredita (npr. 10).

Obično se u prvoj koloni tabele doda još jedan atribut koji na jedinstven način označava slogove tabele. To je tzv. šifra, ili identifikacioni broja (ID broj) sloga, kao na primer, predmetID.

Kolone-atributi

	predmetID	oznakaPredmeta	nazivPredmeta	brojECTS
Redovi-slogovi	1111	CS101	Uvod u objektno-orijentisano programiranje	10
	1112	CS102	Objekti i apstrakcija podataka	10
	1113	CS220	Arhitektura računara	8
	1114	CS232	C/C++ programski jezik	8

Slika 1.1.3 Primer jedne relacije (tabele) sa slogovima - torkama i vrednostima - atributima

VEZE IZMEĐU TABELA

Postoje tabele (npr. Enrollment) čija je osnovna namena da poveže slogove datih i tzv. osnovnim tabelama (npr. Student i Course). To su tabele povezivanja. Veza im je zajednički atribut

Kao što postoji povezanost među redovima jedne tabele, tako može da postoji i povezanost među podacima predstavljeni u različitim tabelama. Veza među njima je zajednički atribut, tj. kolona. Na slikama 4, 5 i 6 dajemo ovde tabele iz udžbenika jer ćemo ih koristiti i kod primera koje ćemo razmatrati, tj. programa koje ćemo kreirati. Na slici 4 prikazana je tabela (relacija) koja sadrži predmete jednog univerziteta sa atributima koji karakterišu svaki predmet. Pored ove tabele, ovde se navode još dve tabele, na slici 5 i 6. Prva sadrži podatke o studentu (te se zove **Student**) a druga tabela sadrži podatke predmetima koje su studenti slušali i položili (ovde nazvana Enrollment).

courseId	subjectId	courseNumber	title	numOfCredits
11111	CSCI	1301	Introduction to Java I	4
11112	CSCI	1302	Introduction to Java II	3
11113	CSCI	3720	Database Systems	3
11114	CSCI	4750	Rapid Java Application	3
11115	MATH	2750	Calculus I	5
11116	MATH	3750	Calculus II	5
11117	EDUC	1111	Reading	3
11118	ITEC	1344	Database Administration	3

Slika 1.1.4 Tabela Course čiji slogovi sadrže podatke o predmetima

Tabele **Student** i **Enrollment** su povezane zajedničkim atributom **ssn**. Na taj način, se zna koji je student polagao neki predmet. A da bi se znalo i koji je predmet polagao, mora se tabela **Enrollment** povezati sa tabelom **Course**

Veza između ove dve tabele je zajednički atribut **courseId**. Na taj način se može utvrditi koji je student polagao neki predmet i koju je ocenu dobio. Očigledno, tabela **Enrollment** povezuje osnovne tabele **Course** i **Student**.

Student Table									
ssn	firstName	mi	lastName	phone	birthDate	street	zipCode	deptID	
444111110	Jacob	R	Smith	9129219434	1985-04-09	99 Kingston Street	31435	BIOL	
444111111	John	K	Stevenson	9129219434	null	100 Main Street	31411	BIOL	
444111112	George	K	Smith	9129213454	1974-10-10	1200 Abercorn St.	31419	CS	
444111113	Frank	E	Jones	9125919434	1970-09-09	100 Main Street	31411	BIOL	
444111114	Jean	K	Smith	9129219434	1970-02-09	100 Main Street	31411	CHEM	
444111115	Josh	R	Woo	7075989434	1970-02-09	555 Franklin St.	31411	CHEM	
444111116	Josh	R	Smith	9129219434	1973-02-09	100 Main Street	31411	BIOL	
444111117	Joy	P	Kennedy	9129229434	1974-03-19	103 Bay Street	31412	CS	
444111118	Toni	R	Peterson	9129229434	1964-04-29	103 Bay Street	31412	MATH	
444111119	Patrick	R	Stoneman	9129229434	1969-04-29	101 Washington St.	31435	MATH	
444111120	Rick	R	Carter	9125919434	1986-04-09	19 West Ford St.	31411	BIOL	

Slika 1.1.5 Tabela Student sadrži attribute studenta

Enrollment Table			
ssn	courseId	dateRegistered	grade
444111110	11111	2004-03-19	A
444111110	11112	2004-03-19	B
444111110	11113	2004-03-19	C
444111111	11111	2004-03-19	D
444111111	11112	2004-03-19	F
444111111	11113	2004-03-19	A
444111112	11114	2004-03-19	B
444111112	11115	2004-03-19	C
444111112	11116	2004-03-19	D
444111113	11111	2004-03-19	A
444111113	11113	2004-03-19	A
444111114	11115	2004-03-19	B
444111115	11115	2004-03-19	F
444111115	11116	2004-03-19	F
444111116	11111	2004-03-19	D
444111117	11111	2004-03-19	D
444111118	11111	2004-03-19	A
444111118	11112	2004-03-19	D
444111118	11113	2004-03-19	B

Slika 1.1.6 Tabela Enrollment povezuje slogove tabela Course i Student

OGRANIČENJA INTEGRITETA

Svako ograničenje integriteta definiše uslov koje vrednosti u nekoj tabeli moraju da zadovolje.

Svako ograničenje integriteta definiše uslov koje vrednosti u nekoj tabeli moraju da zadovolje. Na slici 7 navedene su primeri ograničenja integriteta definisanih za tabele **Subject** i **Course**.

Postoji tri tipa ograničenja:

1. Ograničenja domena
2. Ograničenja primarnog ključa
3. Ograničenja sekundarnog ključa

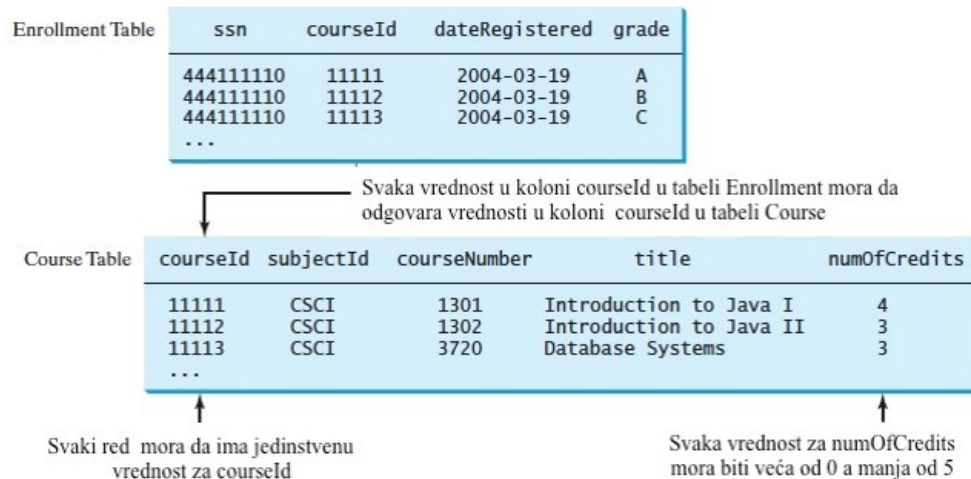
Ograničenja domena i primarnog ključa su intra-relaciona ograničenja jer se odnose samo na jednu relaciju (tabelu), Ograničenje sekundarnog ključa je inter-relaciono, jer se odnosi na više od jedne relacije (tabele).

Ograničenja domena definišu dozvoljene vrednosti nekog atributa. Mogu da se odnose korišćenje nekog od standardnih tipove podataka, a mogu da budu i už, da ograniče opseg njihovih vrednosti.

Ključevi po kojima se mogu povezivati tabele mogu se podeliti i na sledeći način:

1. **Superključ:** atribut ili skup atributa koji na jedinstveni način identifikuje jednu realciju. Skup svih atributa u jednoj relaciji (slogu) je superključ.
2. **Ključ-kandidat:** podskup super ključa, koji može d aima više ključa-kandidata.

Primarni ključ je jedan od ključeva-kandidata, određen pod strane projektanta baze i često se koriste za identifikaciju slogova tabele (npr. `courseId` u tabeli `Course` na slici 4).



Slika 1.1.7 Tabele Enrollment i Course imaju ograničenja integriteta

SEKUNDARNI (STRANI) KLJUČ

Sekundarni (strani) ključ ograničava veze između tabela (relacija). DBMS proverava ispunjenost ograničenja integriteta i odbija da izvrši operacije koje krše ova ograničenja

Podatci su povezani y relacionoj bazi. Slogovi (zapisi, torke, **tuples**) u jednoj relaciji (tabeli) su povezani, a slogovi u različitim tabelama (relacijama) povezani su samo preko zajedničkih atributa, koji se nazivaju sekundarnim (stranim, foreign) ključevima. Sekundarni (strani) ključ ograničava veze između tabela (relacija).

Skup atributa **FK** je strani ključ (foreign key) u relaciji **R** koja povezuje relaciju **T** ako zadovoljava sledeća dva pravila:

1. Atributi **FK** imaju isti domen kao i primarni ključ u **T**.
2. Vrednost **nonnull** za **FK** u **R** mora da odgovara vrednosti primarnog ključa u **T**.

Kao što je pokazano na slici 7, `courseId` je sekundarni (strani) ključ u relaciji **Enrollment** koja povezuje primarni ključ `courseId` u relaciji **Course**. Svaka vrednost za `courseId` mora da odgovara nekoj vrednosti `courseId` u relaciji **Course**.

Sistem za upravljanje bazama podataka (DBMS) proverava ispunjenost ograničenja integriteta i odbija da izvrši operacije koje krše ova ograničenja. Na primer, ako bi pokušali da ubacimo novi slog ("11115," "CSCI," "2490," "C++ Programming," 0) u tabelu **Course**, ta operacija bi bila odbačena jer broj kredita (`numOfCredits`) mora da bude veći od 0. Takođe

bi došlo do odbacivanja operacije ubacivanja novog sloga ako koristi isti primarni ključ kao neki od postojećih slogova.

Ako pokušate da obrišete slog iz tabele **Course** čiji se primarni ključ koristi u slogu tabele **Enrollment**, DBMS bi odbio tu operaciju.

▼ 1.1 Pokazni primeri

PRIMER 1 - KREIRANJE KORISNIKA SISTEMA MYSQL

SQL (Structured Query Language) je jezik kojim se definišu tabele i ograničenja integriteta, i koji omogućava pristup i manipulaciju podataka u tabelama.

SQL je univerzalni jezik za pristupanje relacionim sistemima baza podataka. Krajnji korisnik ne mora da zna i koristi SQL, ako mi aplikacioni program to omogućava. Međutim, taj aplikacioni program mora da koristi SQL (u pozadini) da bi korisnik preko GUI, koristio podatke iz relacione baze podataka.

Mi ćemo ovde izložiti osnove standardnog SQL. Pojedini proizvođači sistema za upravljanje bazama podataka (DBMS), npr. MySQL, Oracle, Sybase, IBM DB2, IBM Informix, MS Access, koriste sopstvena proširenja standardnog SQL jezika, te kada radite sa konkretnim DBMS sistemom, treba da se upoznate sa tim eventualnim proširenjima i specifičnostima.

Prevo ćemo se upoznati sa instalacijom nekod od DBMS sistema, da bi koristili njegov SQL. Pretpostavimo da je to MySQL 5, početne konfiguracije. Da bi koristili primere koje ćemo u ovom objektu učenja iznosititi, treba da koristite korisničko ime: **scott**, sa lozinkom **tiger**.

Administrativne zadatke ćete obavljati koristeći MySQL Workbench , a možete koristiti i komandnu liniju računara. MySQL Workbench je GUI koji upravlja MySQL bazom podataka. Ovde dajemo korake kreiranja jedne baze podataka, pri čemu ćemo koristiti komandnu liniju.

1. Pored upitnog znaka DOS-a, na komandnoj liniji, treba da ukucate :

mysql -uroot -p

2. Na upitni znak mysql, treba da ukucate:

use mysql;

3. Kreiraje korisnika scott sa lozinkom: tiger:

create user 'scott'@'localhost' identified by 'tiger';

4. Da bi korisniku scott dali odobrenje za razne SQL operacije, treba da otkucate:

grant select, insert, update, delete, create, create view, drop, execute, references on *.* to 'scott'@'localhost';

Ako želite da mu dodeliti pristup sa neke udaljene IP adrese, otkucajte:

grant all privileges on *.* to 'scott'@'%' identified by 'tiger';

Ako želite da udeljeni pristup može biti samo sa određene IP adrese, treba da otkucate:

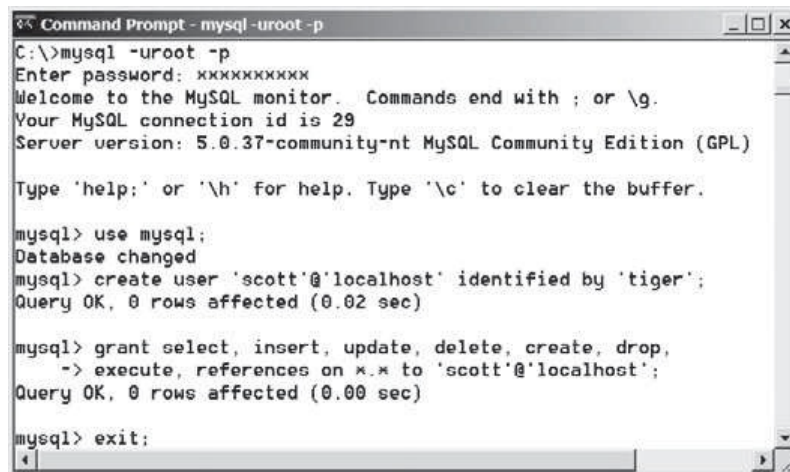
grant all privileges on *.* to 'scott'@'ipAddress' identified by 'tiger';

5. Otkucajte: ***exit***

PRIMER 2 - KREIRANJE BAZE PODATAKA

MySQL server koristi dve baze podataka pod nazivima: mysql i test. Baza mysql je namenjena administratoru servera, a test baza može da se koristi za smeštaj tabela korisnika.

Na slici 1 prikazan je prikaz vaše sesije kreiranja korisnika MySQL sistema na prozoru Windows OS koji omogućava rad sa komandnom linijom.



```
Command Prompt - mysql -uroot -p
C:\>mysql -uroot -p
Enter password: xxxxxxxxxxxx
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 29
Server version: 5.0.37-community-nt MySQL Community Edition (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use mysql;
Database changed
mysql> create user 'scott'@'localhost' identified by 'tiger';
Query OK, 0 rows affected (0.02 sec)

mysql> grant select, insert, update, delete, create, drop,
-> execute, references on *.* to 'scott'@'localhost';
Query OK, 0 rows affected (0.00 sec)

mysql> exit;
```

Slika 1.2.1 Pristup sistemu MySQL preko komandne linije

Opaska: Pri radu sa Windows OS, MySQL server startuje uvek kada uključite kompjuter. Ovo možete yaustaviti ako otkucate na komandnoj liniji: ***net stop mysql***

Ako želite da vratite poečtni način rada, treba da otkucate: ***net start mysql***

MySQL server koristi dve baze podataka pod nazivima: ***mysql*** i ***test***. Baza podataka ***mysql*** sadrži tabele koje sadrže informacije o serveru i njegovim korisnicima. Ova baza je namenjena administratoru servera. Kako je MySQL instalisan na vašem računaru, vi imate pristup ovoj bazi. Međutim, ne bi trebalo da koristite ovu bazu za smeštaj tabela korisnika MySQL sistema. Za tu svrhu treba da koristite ***test*** bazu. Možete i da kreirate novu bazu upotrebom komande za kreiranje baze podataka:

create database databasename

Ako želite da obrišete ranije kreiranu bazu, onda koristite naredbu:

delete databe databasename

Naravno, na mesto databasename treba da stavite nayiv vaše baze podataka.

PRIMER 3 - KREIRANJE BAZE PODATAKA

*Umesto da uvek kucate SQL naredbe, možete ih jedanput uneti u neku datoteku, npr. **script.sql** i onda je memorisati za kasniju upotrebu.*

Kreiraćemo bazu podataka pod nazivom **javabook** koju ćemo koristiti u svim naredim primerima. Primenićemo sledeće korake u kreiranju ove baze:

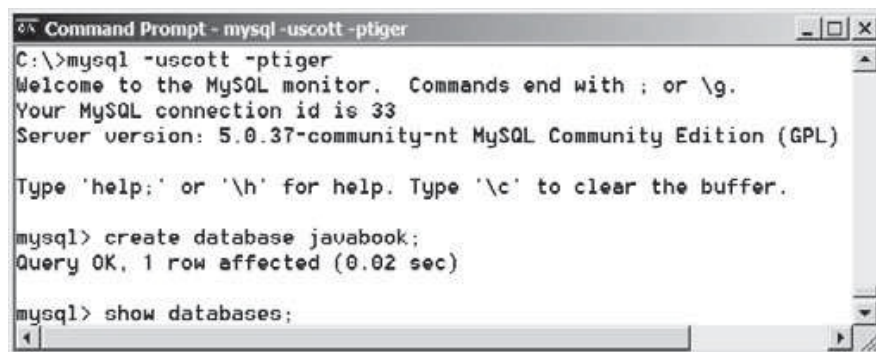
1. U DOS komandnoj liniji otkucajte:

mysql -uscott -ptiger

čime se uključujemo u masql, kao što je prikazano na slici 2.

2. Bazu javabook kreira sledećom naredbom:

create database javabook;



```
Command Prompt - mysql -uscott -ptiger
C:\>mysql -uscott -ptiger
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 33
Server version: 5.0.37-community-nt MySQL Community Edition (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> create database javabook;
Query OK, 1 row affected (0.02 sec)

mysql> show databases;
```

Slika 1.2.2 Primer kreiranja baze javabook

Umesto da uvek kucate SQL naredbe, možete ih jedanput uneti u neku datoteku, npr. **script.sql** i onda je memorisati za kasniju upotrebu.

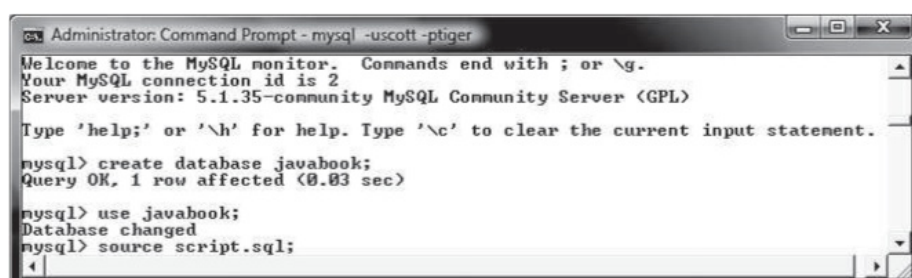
Da bi tu skript datoteku koristili, treba prvo da otkucate:

use javabook

a onda:

source script.sql;

kao što je pokazano na slici 3.



```
Administrator: Command Prompt - mysql -uscott -ptiger
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.1.35-community MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database javabook;
Query OK, 1 row affected (0.03 sec)

mysql> use javabook;
Database changed
mysql> source script.sql;
```

Slika 1.2.3 Izvršavanje SQL komande primenom script datoteke

PRIMER 4 - KREIRANJE TABELE COURSE

Naredba create kreira tabelu u bazi podataka. Pored naziva tabele, potrebno je navesti sve njene attribute i njihove tipove podataka.

Da bi kreirali tabelu, treba koristiti naredbu create i da navedete njen naziv, njene attribute i tipove. Daćemo primer kreiranja tabele Course na slici 5 i 6:

courseId	subjectId	courseNumber	title	numOfCredits
11111	CSCI	1301	Introduction to Java I	4
11112	CSCI	1302	Introduction to Java II	3
11113	CSCI	3720	Database Systems	3
11114	CSCI	4750	Rapid Java Application	3
11115	MATH	2750	Calculus I	5
11116	MATH	3750	Calculus II	5
11117	EDUC	1111	Reading	3
11118	ITEC	1344	Database Administration	3

Slika 1.2.4. Tabela Course

```
create table Course (  
  courseId char(5),  
  subjectId char(4) not null,  
  courseNumber integer,  
  title varchar(50) not null,  
  numOfCredits integer,  
  primary key (courseId)  
);
```

Ovaj iskaz kreira tabelu **Course** sa atributima: **courseId**, **subjectId**, **courseNumber**, **title**, i **numOfCredits**.

Svaki atribut ima svoj tip podataka koji definiše tip podataka koji sme da se meoriše kao atribut, Na primer,

- **char(5)** određuje da se vrednost za courseId sastoji od pet znakova.
- **varchar(50)** određuje da je naziv string promenljive dužine, ali sa najviše 50 znakova
- **Integer** specificira da je vrednost atributa courseNumber mora da bude ceo broj.

Takođe je navedeno da je primarni ključ označen sa **courseId**.

```

Command Prompt - mysql-uscott-ptiger
mysql> use javabook;
Database changed
mysql> drop table Course;
Query OK, 0 rows affected (0.08 sec)

mysql> create table Course(
->   courseId char(5),
->   subjectId char(4) not null,
->   courseNumber integer,
->   title varchar(50) not null,
->   numOfCredits integer,
->   primary key (courseId)
-> );
Query OK, 0 rows affected (0.11 sec)

mysql>

```

Slika 1.2.5 Primer kreiranja tabele Course

PRIMER 5 - KREIRANJE TABELA STUDENT I ENROLLMENT

Primarni ključ se specificira svojom oznakom, tj. nazivom atributa koji se koristi kao primarni ključ. Strani ključ se specificira navođenjem njegove veze sa tabelom u kojoj je on primarni ključ

Tabela **Student** (slika 6) kreira se sledećim SQL iskazom:

Enrollment Table			
ssn	courseId	dateRegistered	grade
444111110	11111	2004-03-19	A
444111110	11112	2004-03-19	B
444111110	11113	2004-03-19	C
444111111	11111	2004-03-19	D
444111111	11112	2004-03-19	F
444111111	11113	2004-03-19	A
444111112	11114	2004-03-19	B
444111112	11115	2004-03-19	C
444111112	11116	2004-03-19	D
444111113	11111	2004-03-19	A
444111113	11113	2004-03-19	A
444111114	11115	2004-03-19	B
444111115	11115	2004-03-19	F
444111115	11116	2004-03-19	F
444111116	11111	2004-03-19	D
444111117	11111	2004-03-19	D
444111118	11111	2004-03-19	A
444111118	11112	2004-03-19	D
444111118	11113	2004-03-19	B

Slika 1.2.6 Tabela Student

```

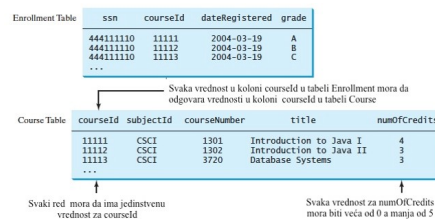
create table Student (
  ssn char(9),
  firstName varchar(25),

```



```
mi char(1),
lastName varchar(25),
birthDate date,
street varchar(25),
phone char(11),
zipCode char(5),
deptId char(4),
primary key (ssn)
);
```

Tabela **Enrollment** (slika 7) kreira se sledećim SQL iskazom:



Slika 1.2.7 . Tabela Enrollment

```
create table Enrollment (
  ssn char(9),
  courseId char(5),
  dateRegistered date,
  grade char(1),
  primary key (ssn, courseId),
  foreign key (ssn) references Student(ssn),
  foreign key (courseId) references Course(courseId)
);
```

PRIMER 6 - BRISANJE TABELE

*Komandom **drop table** sa komandne linije, briše se navedena tabela*

Ako tabela više nije potrebna, može se obrisati za stalno upotrebom komande **drop table**. Na primer:

drop table Course;

Ako se tabela koju brišete poziva iz drugih tabela, onda morate prvo da obrišete te druge tabele da obrišete. Na primer, ako imamo kreirane tabele **Course**, **Student** i **Enrollment**, i ako želimo da obrišemo tabelu **Course**, moramo da obrišemo prvo tabelu **Enrollment**, jer ona poziva tabelu **Course**.

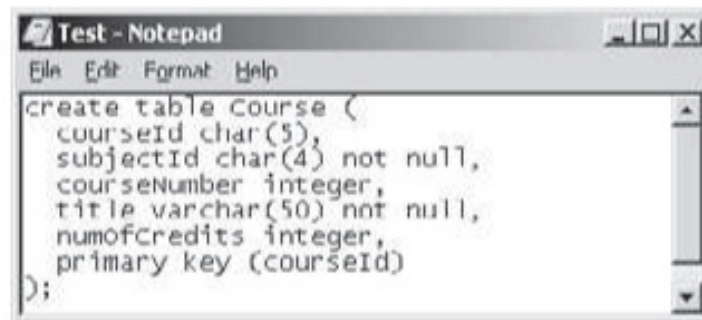
Ako napravite grešku u kucanju iskaza, trebalo bi da ga ponovo otkucate. Da bi se ovo izbeglo, možete memoristai iskaz u nekoj datoteci i onda da izvršite taj iskaz iz te datoteke. Da bi se

ovo uradilo, treba da kreirate tekstualnu datoteku, na primer, test.sql korišćenjem bilo kog tekstualnog editora (npr. Notepad), kao što je prikazano na slici 8a.

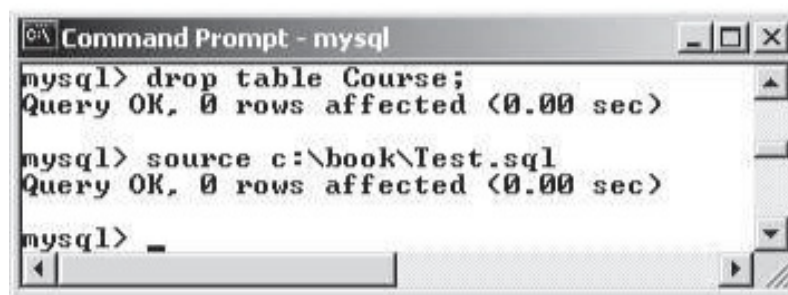
Da bi dali komentar u nekoj liniji, treba da linija počinje sa dve kose crte. Možete izvršiti ovaj skript kucanjem :

source test.sql

sa SQL komandne linije, kao što je pokazano na slici 8b.



a)



b)

Slika 1.2.8 Primer kreiranja iz skripta i brisanja tabele Course

PRIMER 7 - UNOS, MENJANJE I BRISANJE PODATAKA IZ TABELE

Ključne reči za unos je insert, za menjanje je update, a za brisanje je delete. Ove operacije se mogu da primene na jednoj vrednosti atributa, na celom slogu, ili svim slogovima tabele.

Kada je tabela kreirana, možete da unosite podatke u nju, kao što možete i da ih menjate ili brišete. Sintaksa za unos podataka u tabelu je sledeća:

```
insert into tableName [(column1, column2, ..., column)]  
  values (value1, value2, ..., valuen);
```

Na primer, sledeći iskaz unosi slog podataka u tabelu Course. Slog unosi sledeće podatke: courseId '11113', subjectId 'CSCI', courseNumber '3720', title 'Database Systems', and creditHours 3.


```
insert into Course (courseId, subjectId, courseNumber, title, numOfCredits)
values ('11113', 'CSCI', '3720', 'Database Systems', 3);
```

Nazivi kolona ne moraju da se obavezno navedu. Ako se ne navode, onda se moraju uneti sve vrednosti kolona (atributa) sloga. Vrednosti stringova su osetljive na veličinu slova i moraju se navesti između kvota sa jednom crtom.

Sintaksa za promenu u nekoj tabeli je:

```
update tableName
set column1 = newValue1 [, column2 = newValue2, ...]
[where condition];
```

Na primer, sledeći iskaz menja vrednost za **numOfCredits** za predmet čije je naziv **Database Systems** i nova vrednost je 4.

```
update Course
set numOfCredits = 4
where title = 'Database Systems';
```

Sintaksa za brisanje sloga iz tabele je:

```
delete from tableName
[where condition];
```

Na primer, sledeći iskaz briše predmet Database Systems iz tabele Course:

```
delete from Course
where title = 'Database Systems';
```

Sledeći iskaz briše sve slogove iz tabele Course:

```
delete from Course;
```

PRIMER 8 - JEDNOSTAVNI UPITI BAZE

Select izdvaja navedene kolone, a from definiše tabele koje se koriste, a where definiše uslove koje redovi tabele (slogovi) treba da zadovolje da bi bili izabrani.

Da bi se dobili podaci sa tabela, koristi se select iskaz sa sledećom sintaksom:

```
select column-list
from table-list
```

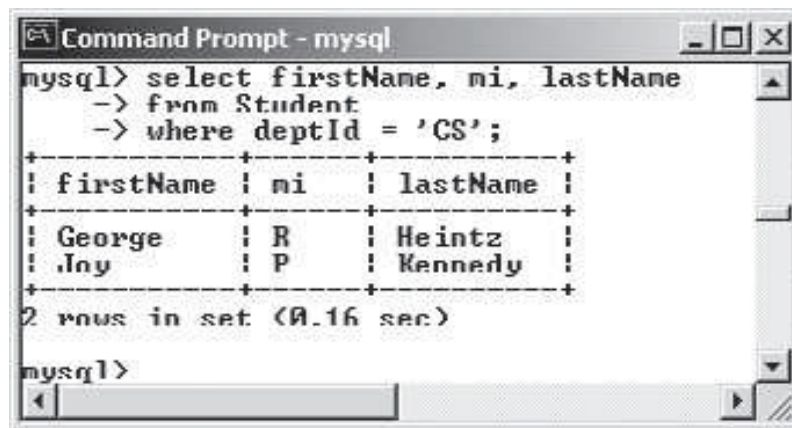
```
[where condition];
```

Iza select ključne reči, nalazi se kolona vrednosti koja se bira, a iza ključne reči from nalazi se lista tabeli koje su uključene u upit. opcioni where iskaz specificira uslove za izabrane redove tabela.

Navešćemo nekolikoprimeru upita.

Upit 1: Izaberite sve studente CS odseka, kao što je prikazano na slici 9.

```
select firstName, mi, lastName
from Student
where deptId = 'CS';
```



Slika 1.2.9 Primer select upita

PRIMER 9 - UPOREĐENJA I LOGIČKE OPERACIJE

Operatori upoređenja i logički operatori se koriste u iskazu where koji daje uslove koje moraju da zadovolje redovi tabela, a da bi se izdvojili.

SQL ima šest operatora upoređenja i tri logička operatora, prikazani u tabelama 1 i 2.

Tabela 1: Operatori upoređenja

Operator	Opis
=	Jednako
<> ili !=	Nije jednako
<	Manje
<=	Manje ili jednako
>	Veće
>=	Veće ili jednako

Tabela 2: Logički operatori

Operator	Opis
not	nije
and	i
or	ili

Upit 2: Izdvojite imena studenta koji su u CS odseku i žive u mestu sa poštanskim brojem (ZIP) 31411

```
select firstName, mi, lastName
from Student
where deptId = 'CS' and zipCode = '31411';
```

Ako se želi da se izdvoje svi atributi iz neke tabele, umesto navođenja njihovih imena, dovoljno je navesti zvezdicu * iza select iskaza. Na primer, sledeći upit prikazuje sve attribute studenta iz CS odseka i sa poštanskog broja 3144.

```
select *
from Student
where deptId = 'CS' and zipCode = '31411';
```

PRIMER 10 - OPERATORI LIKE, BETWEEN-AND I IS NULL

SQL operatori like, beetween-end i null omogućavaju nalaženje željene kombinacije podataka u tabelama.

SQL ima operatore like koji se koristi za nalaženje određenih kombinacija podataka u bazama. Sintaksa provere da li string **s** ima kombinaciju oznaka **p** je sledeća:

s like p or s not like p

Može da koristiti i džokera za oznake % i _ u kombinaciji **p**. Džoker oznaka % predstavlja *nule ili više oznaka*, a džoker oznaka _ odgovara *bilo kojoj pojedinačnoj oznaci* u stringu **s**.

Na primer,

- **lastName like '_mi%'** odgovara bilo kom stringu čije drugo i treće slovo **m** i **i**.
- **lastName not like '_mi%'** isključuje bilo koji string čije drugo i treće slovo su **m** i **i**

Operator **between-and** proverava da li vrednost **v** je između dve druge vrednosti **v1** i **v2** , upotrebom sintakse:

v between v1 and v2 or v not between v1 and v2

Dodatno objašnjenje:

- **v between v1 and v2** je ekvivalentno sa **v >= v1 and v <= v2**,
- **v not between v1 and v2** je ekvivalentno sa **v < v1 or v > v2**

Operator **null** proverava da li je neka vrednost **v** jednaka **null** upotrebom sledeće sintakse.

v is null or v is not null

Upit 3: Nađi brojeve socijalnog osiguranja (ssn) studenata koji imaju ocene između 'C' i 'A'.

```
select ssn  
from Enrollment  
where grade between 'C' and 'A';
```

PRIMER 11 - SINONIMI OZNAKA KOLONA

SQL predviđa mogućnost da dodelite sinonime oznakama kolona, tj. alternativne, duže nazive koji se koriste kod prikazivanja rezultata upita. Komanda je as

Kod prikazivanja rezultata upita, SQL koristi nazive kolona kao njihova zaglavlja. Kako programeri obično označavaju komone (atribute) skraćenim oznakama, i kako kolenemešusobno nemaju prazan prostor, to se pri prikazu kolona dobijenih kao rezultat SQL upita ne dobija dovoljno pregledna slika rezultata. Ako bi se koristili duža, opisna imena kolona u njihovim zaglavlja, ondabi se poboljšala preglednost, jer bi oznake kolona bile razumljivije, a stvorio bi se i razmak između kolona.

Da bi se ovo ostvarilo, SQL predviđa mogućnost da dodelite sinonime oznakama kolona, tj. alternativne, duže nazive koji se koriste kod prikazivanja rezultata upita. Sinonimi oznaka kolona se mogu definisati sledećom sintaksom:

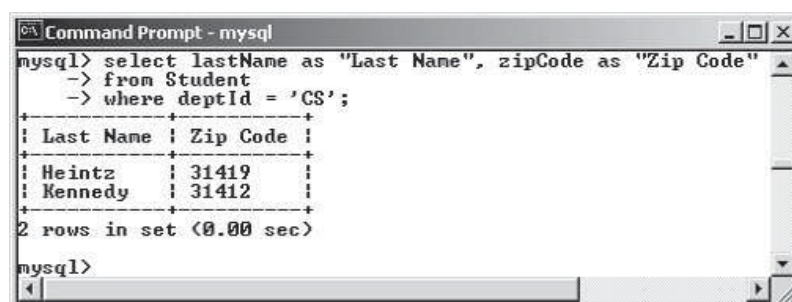
select **columnName** [as] alias

Upit 4: Izdvojite prezime i poštanski broj studenata u CS odseku. Prikažite zaglavlja kolona ka "Last Name" umesto oznake lastName i "ZIP Code" umesto oznake zipCode.

SQL upit je sledeći:

```
select lastName as "Last Name", zipCode as "Zip Code"  
from Student  
where deptId = 'CS';
```

Na slici 11 prikazan je dobijen rezultat na monitoru računara:



Slika 1.2.10 Primena sinonima u zaglavlja kolona

PRIMER 12 - ARITMETIČKI OPERATORI

*Možete koristiti uobičajene aritmetičke operatore * (množenje), / (division), + (sabiranje), i - (oduzimanje) u SQL upitima.*

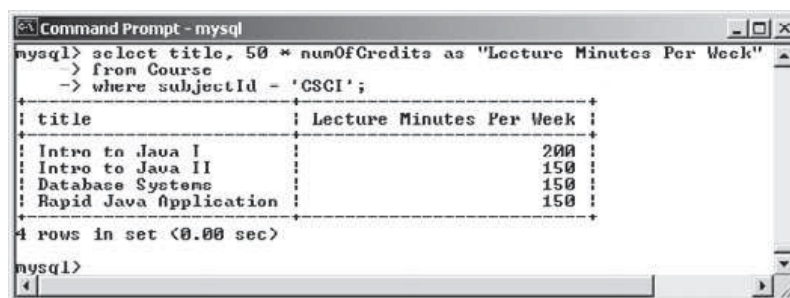
Možete koristiti uobičajene aritmetičke operatore * (množenje), / (deljenje), + (sabiranje), i - (oduzimanje) u SQL upitima (**queries**).

Upit 5: Ako pretpostavimo da jedan kredit na predmetu odgovara jednom času od 50 minuta predavanja, odredite ukupan broj minuta svakog predmeta nedeljno koji se odnosi na CSCI.

SQL upit izgleda ovako:

```
select title, 50 * numOfCredits as "Lecture Minutes Per Week"
from Course
where subjectId = 'CSCI';
```

Na slici 12 prikazan je dobijen rezultat na monitoru.



The screenshot shows a MySQL Command Prompt window with the following content:

```
mysql> select title, 50 * numOfCredits as "Lecture Minutes Per Week"
-> from Course
-> where subjectId = 'CSCI';
```

title	Lecture Minutes Per Week
Intro to Java I	200
Intro to Java II	150
Database Systems	150
Rapid Java Application	150

4 rows in set (0.00 sec)

```
mysql>
```

Slika 1.2.11 Upotreba aritmetičkih operatora u SQL upitima

PRIMER 13 - PRIKAZIVANJE SLOGOVA TABELA BEZ DUPLIRANJA

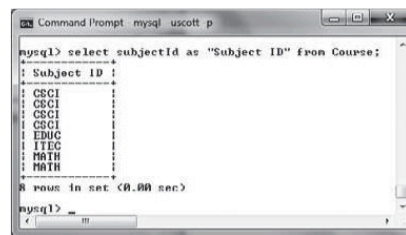
*SQL obezbeđuje ključnu reč **distinct** koja se može da koristi za eliminisanje dupliciranih slogova (redova) u tabelama*

SQL obezbeđuje ključnu reč **distinct** koja se može da koristi za eliminisanje dupliciranih slogova (redova) u tabelama.

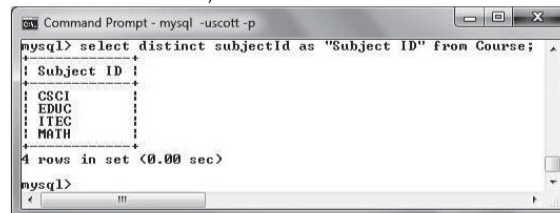
Primer SQL upita sa **distinct** kojim se eliminišu duplirani slogovi:

```
select distinct subjectId as "Subject ID"
from Course;
```

Slika 13a prikazuje sve ID oznake predmeta, a slika 13b prikazuje samo različite ID oznake predmeta, dobijeni gornjim upitom.



a)

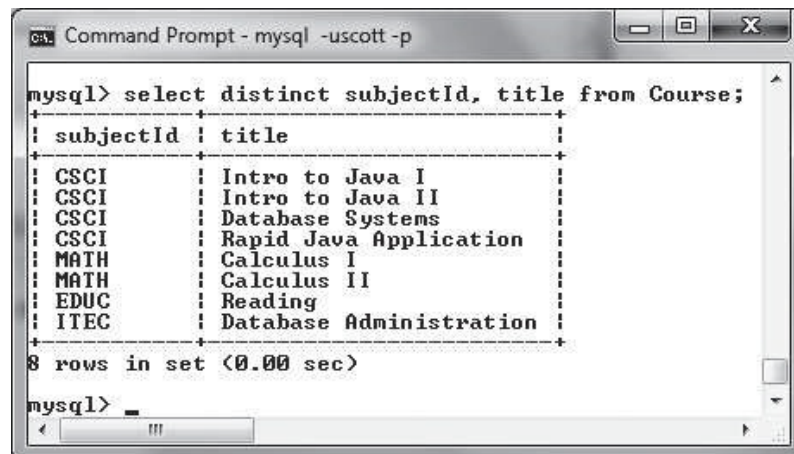


b)

Slika 1.2.12 a) Prikaz sa duplim slogovima b) Prikaz bez duplih slogova

Kada ima više od jedne kolone u select iskazu, ključna reč **distinct** se odnosi na sve slogove u rezultatu. Na primer, sledeći iskaz prikazuje sve slogove sa **distinct subjectId** i **title** kao što je prikazano na slici 14. Pojedini slogovi mogu da imaju iste vrednosti za **subjectId**, ali različite vrednosti za **title**. Zato su ovi slogovi različiti, te se prikazuju u dobijenom rezultatu na slici 14. Rezultat je dobijen primenom sledećeg upita:

```
select distinct subjectId, title
from Course;
```



Slika 1.2.13 Ključna reč distinct se odnosi na ceo slog

PRIMER 14 - PRIKAZ SORTIRANIH SLOGOVA

*Ključna reč **order by** u SQL select iskazu sortira dobijene slogove*

SQL omogućava sortiranje slogova u dobijenom rezultatu primenom ključne reči **order**. Za to se koristi sledeća sintaksa:

```
select column-list  
  from table-list  
  [where condition]  
  [order by columns-to-be-sorted];
```

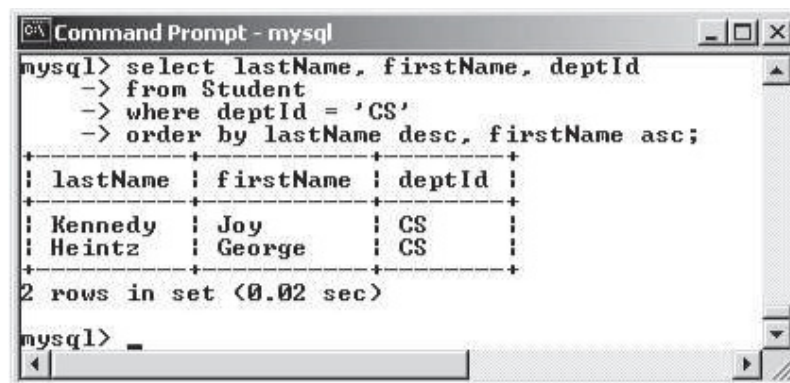
U ovoj sintaksi, columns-to-be-sorted specificira kolonu ili listu kolona koja se sortira. Po pravilu, sortiranje je *po rastućem redosledu*. Ako se želi sortiranje *po opadajućem redu*, treba koristiti ključnu reč **desc**. Možete koristiti i ključnu reč **asc** posle kolone **columns-to-be-sorted**, ali *to nije potrebno*.

Kada je navedeno više kolona za sortiranje, redovi se sortiraju prvo prema prvoj koloni, pa onda redovi kod koji se ponavljaju vrednosti u prvoj koloni, sortiraju se prema drugoj koloni itd.

Upit 6: Izlistaj ime i prezime studenata u CS odseku, poređanih prvo prema svojim prezimenima u opadajućem redosledu a onda prema imenima i to prema rastućem redosledu.

Upit izgleda ovako, a rezultat je prikazan na slici 15 :

```
select lastName, firstName, deptId  
  from Student  
 where deptId = 'CS'  
 order by lastName desc, firstName asc;;
```



Slika 1.2.14 Možete sortirati slogove primenom iskaza order by

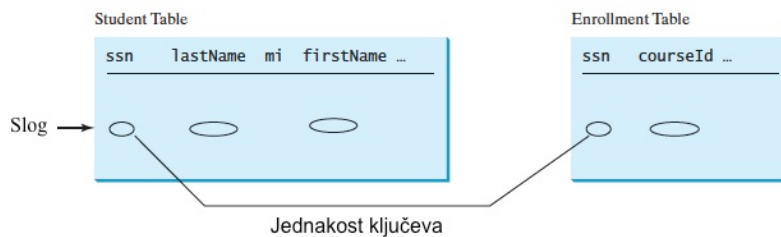
PRIMER 15 - SPAJANJE TABELA

Spajanjem tabela preko zajedničkih kolona, mogu se dobiti podaci iz više od jedne tabele. Iskaz where određuje uslove spajanja.

Često je potrebno da se dobiju podaci iz više tabela, kao što je pokazanu u sledećem upitu:

Upit 7: Izlistaj predmete studenta Jacob Smith.

Da bi rešili ovaj upit, potrebno je da spojite tabele Student i Enrollment, kao što je pokazano na slici 16.

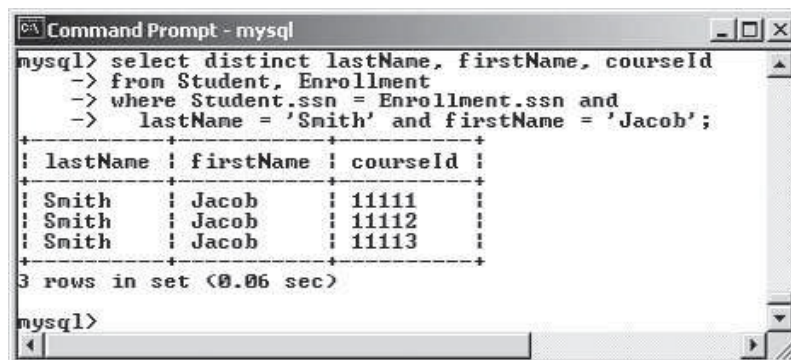


Slika 1.2.15 Tabele Student i Enrollment su spojene preko kolone ssn

Upit se rešava sledećim SQL upitom:

```
select distinct lastName, firstName, courseId
from Student, Enrollment
where Student.ssn = Enrollment.ssn and
lastName = 'Smith' and firstName = 'Jacob';
```

Tabele **Student** i **Enrollment** su navedene u iskazu **from**. Upit ispituje svaki par redova, koji čini red iz tabele **Student** i iz tabele **Enrollment**. Upit bira par koji zadovoljava uslov dat u iskazu **where**. Redovi u tabeli **Student** imaju atribut za prezime **Smith**, i ime **Jacob**, i oba reda iz tabela **Student** i **Enrollment** imaju iste vrednosti atributa **ssn**. Za svaki izabrani red, koriste se atributi **lastName** i **firstName** iz tabele **Student**, a **courseId** iz tabele **Enrollment**. Na slici 17 je prikazan dobijeni rezultat. Tabele **Student** i **Enrollment** imaju isti atribut **ssn**. Da bi se međusobno razlikovali u upitu, upotrebljavamo: **Student.ssn** i **Enrollment.ssn**.



Slika 1.2.16 Rezultat upita u kome se povezuju dve tabele

1.2 Zadaci za samostalni rad

ZADACI ZA SAMOSTALNI RAD STUDENTA

Cilj je provežbavanje pisanja SQL upita

Zadatak 1:

U svaku od tabela iz pokaznih primera dodati po jedan red korišćenjem naredbe insert.

Zadatak 2:

Izlistaj sve predmete studenta sa imenom Mark i prezimenom koje počinje slovom S, a čiji je ssb broj veći od 2000.

Zadatak 3:

Svim studentima koji imaju predmet sa oznakom koja počinje sa CS definisati da broj bodova iznosi 10 (numOfCredits).

▼ Poglavlje 2

JDBC

ŠTA JE JDBC?

JDBC je Java API sa pristup relacionim bazama podataka.

JDBC je **Java API** koji se koristi razvoj Java aplikacija koje koriste relacione baze podataka. **JDBC** obzbeđuje programerima u Javi jedan uniformni interfejs za pristup i za rad sa relacionim bazama podataka. Upotrebom **JDBC API**, aplikacije pisane u Javi mogu da izvrše SQL iskaze, da dobijaju rezultate, da predstavljaju podatke na način koji je ugodan korisniku i da vrate promenjene podatke u bazu podataka. **JDBC API** takođe omogućava da se radi sa više izvora podataka u distribuiranim heterogenim okruženjima.

Veze između Java programa, **JDBC API**, **JDBC** uslužnih programa (**drivers**) i relacionih baza podataka prikazane su na slici 1. **JDBC API** je skup Java interfejsa i klasa koje se upotrebljavaju kod pisanja Java programa koji pristupaju i manipulišu sa relacionim bazama podataka. Kako **JDBC uslužni program (driver)** služi kao interfejs koji olakšava komunikaciju između **JDBC** i neke posebne baze podataka, **JDBC uslužni programi** se razvijaju za specifične baze podataka, tj. **DBMS**, te ih najčešće nude prodavci tih sistema baza podataka. Na primer, vama je potreban **MySQL JDBC** uslužni program da bi pristupili **MySQL** bazi podataka, a **Oracle JDBC** uslužni programi da bi pristupili Oracle bazama podataka. Za **Access** baze podataka, upotrebite **JDBC-ODBC mostni uslužni program** koji je uključen u JDK.

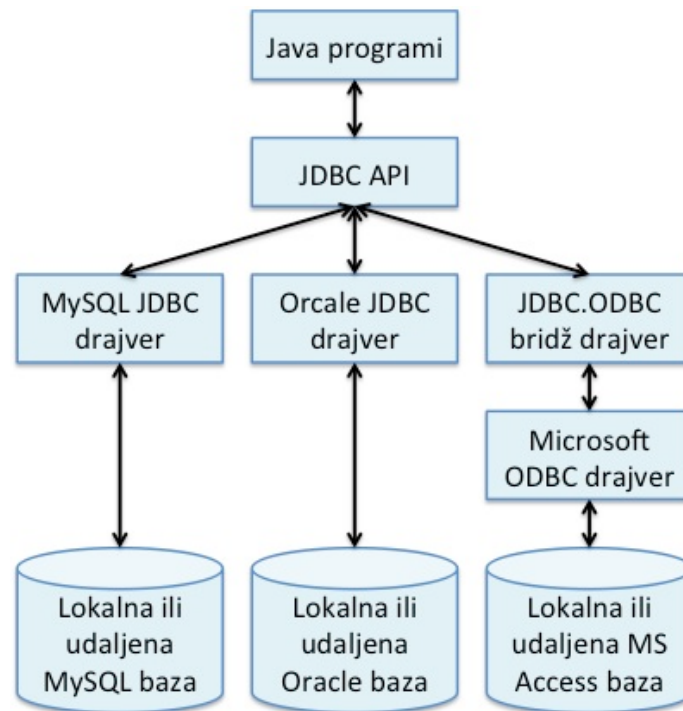
✓ Point Check

Key

Point

FIGURE 32.20

ODBC uslužni program je instalisan u Windows operativnom sistemu. **JDBC-ODBC mostni uslužni program** dozvoljava da Java program pristupi ODBC izvoru podataka, tj. izvoru podataka koji ima pristup preko ODBC uslužnog programa.



Slika 2.1.1 Java program koristi JDBC API da bi pristupio i radio sa relacionom bazom podataka.

RAZVOJ APLIKCIJA SA BAZAMA PODATAKA UPOTREBOM JDBC

*Četiri ključna interfejsa neophodna za razvoj aplikacije sa bazom podataka su: **Driver** , **Connection** , **Statement** , i **ResultSet** .*

JDBC API je interfejs Java plikativnim programima generičkim SQL bazama podataka koji omogućava Java programerima da razvijaju Java aplikacije nezavisne od DBMS, primeni+om uniformnog interfejsa.

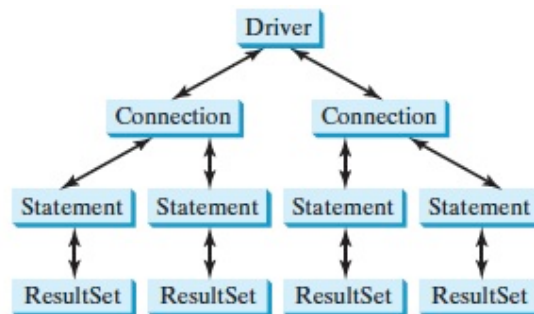
JDBC API sadrži klase i interfejse za

- uspostavljanje veza sa bazama podataka,
- slanje SQL iskaza bazamapodataka,
- obradu rezultata SQL iskaza, i za
- dobijanje metapodataka o bazi podataka.

Četiri ključna interfejsa neophodna za razvoj aplikacije sa bazom podataka su: Driver , Connection , Statement , i ResultSet . Ovi interfejsi predstavljaju radni okvir za pristup generičkim SQL bazama podataka. JDBC API definiše ove interfejse a proizvođači JDBC uslužnih programa (drajvera) obezbeđuju primenu specifikacija ovih interfejsa. Programeri upotrebljavaju ove interfejse.

Veze među ovim interfejsima su prikazane na slici 2. JDBC aplikacija koristi odgovarajući JDBC uslužni program (drajver) upotrebom interfejsa Driver, povezuje se sa bazom podataka upotrebom interfejsa Connection, kreira SQL iskaz

upotrebom interfejsa Statement, i obrađuje rezultate upotrebom interfejsa ResultSet, ako ih iskazi obezbeđuju.



Slika 2.1.2 JDBC interfejsi i njihove međusobne veze

JDBC interfejsi i klase su razvojni blokovi kod razvoja Java baza podataka. Tipičan Java program primenjuje sledeće korake pristupanja bazi podataka:

1. Instalisanje JDBC uslužnih programa (drajvera)
2. Uspostavljanje veze
3. Kreiranje SQL iskaza
4. Izvršavanje SQL iskaza
5. Obrada ResultSet

PRUZIMANJE JDBC USLUŽNIH SOFTVERA (DRAJVERA)

Odgovarajući JDBC uslužni program (drajver) se preuzima od njegovih proizvođača.

Odgovarajući JDBC uslužni program (drajver) se preuzima upotrebom sledećeg iskaza, i to pre povezivanja aplikacije sa bazom podataka.

```
Class.forName("JDBCClass");
```

Drajver je konkretna klasa koja primenjuje java.sql.Driver interfejs. Drajveri za Access, MySQL, i Oracle su dati u sledećoj tabeli. Ako vaš program pristupa nekoliko različitih baza podataka, morate za svaku od njih da instalirate odgovarajući drajver.

Tabela 1: Klase drajvera

DBMS	Klasa drajvera	Izvor
Access	sun.jdbc.odbc.JdbcOdbcDriv	JDK
MySQL	com.mysql.jdbc.Driver	mysql-connector-java-5.1.26.jar
Oracle	oracle.jdbc.driver.OracleDriver	ojdbc6.jar

Slika 2.1.3 JDBC drajveri (uslužni programi)

JDBC-ODBC drajver za Access ugrađen u JDK. Poslenja nezavisna platforma za MySQL JDBC drajver je **mysql-connector-java-5.1.26.jar** i može se preuzeti sa dev.mysql.com/

downloads/connector/j/. Poslednja verzija [Oracle JDBC drajvera](http://www.oracle.com/technetwork/database/enterprise-edition/jdbc-112010-090769.html) je **ojdbc6.jar** (preuzima se www.oracle.com/technetwork/database/enterprise-edition/jdbc-112010-090769.html).

Da bi koristili MySQL i Oracle drajvere, potrebno je da dodate **mysql-connector-java-5.1.26.jar** i **ojdbc6.jar** u **classpath** upotrebom sledeće DOS komande u Windows OS:

:

```
set classpath=%classpath%;  
c:\book\lib\mysql-connector-java-5.1.26.jar;  
c:\book\lib\ojdbc6.jar
```

Ako koristite IDE, kao što su Eclipse ili NetBeans, potrebno je da dodate ove jar datoteke u library ovih razvojnih alata.

Poslednje verzije Jave omogućavaju automatsko instalisanje JDBC drajvera.

USPOSTAVLJANJE VEZE

*Povezivanje baze podataka se vrši pozivanjem statičkog metoda **getConnection(databaseURL)** u klasi **DriverManager***

Povezivanje baze podataka se vrši pozivanjem statičkog metoda **getConnection(databaseURL)** u klasi **DriverManager** na sledeći način:

```
Connection connection = DriverManager.getConnection(databaseURL);
```

gde je **databaseURL** jedinstvena identifikacija baze podataka na Internetu. U donjoj tabeli su dati URL lokacije za Access, MySQL i Oracle baze podataka.

DBMS	URL
Access	jdbc:odbc:dataSource
MySQL	jdbc:mysql://hostname/dbname
Oracle	jdbc:oracle:thin:@hostname:port#:oracleDBSID

Slika 2.1.4 URL za JDBC raznih DBMS

Za ODBC izvor podataka, databaseURL je **jdbc:odbc:dataSource**. ODBC izvor podataka može da se kreira upotrebom **ODBC Data Source Administrator** na Windows OS.

Za slučaj izvora podataka **ExampleMDBDataSource** kreiranog za Access bazu, sledeći iskaz kreira **Connection** objekat:

```
Connection connection = DriverManager.getConnection  
("jdbc:odbc:ExampleMDBDataSource");
```

Argument **databaseURL** za MySQL bazu podataka specificira ime računara i ime baze podataka radi lociranja baze. Na primer, sledeći iskaz kreira Connection objekat za lokalnu MySQL bazupodataka javabook sa korisničkim imenom (username) scott i lozinkom (password) tiger:

```
Connection connection = DriverManager.getConnection
("jdbc:mysql://localhost/javabook", "scott", "tiger");
```

Argument databaseURL za Oracle bazu podataka specificira ime računara (hostname) broj priključka (port#) gde baza osluškuje dolazeće zahteve za bazom, i oracleDBSID naziv baze da bi locira bazu podataka. Na primer, sledeći iskaz kreira Connection objekat za Oracle bazu podataka na liang.armstrong.edu sa username scott i password tiger:

```
Connection connection = DriverManager.getConnection
("jdbc:oracle:thin:@liang.armstrong.edu:1521:orcl",
"scott", "tiger");
```

KREIRANJE I IZVRŠENJE ISKAZA

*Statement object definiše SQL iskaze koje DBMS treba da izvrši i da vrati dobijeni rezultat. SQL upit se može izvršiti upotrebom **executeQuery(String sql)**.*

Connection objekat povezuje vaš progra, i bazu podataka (preko DBMS). Po kreiranju Connection objekta, možete kreirati iskaze za izvršenje SQL iskaza.

Statement object definiše SQL iskaze koje DBMS treba a izvrši i da vrati dobijeni rezultat.

:

```
Statement statement = connection.createStatement();
```

SQL jezik za definisanje podataka (DDL-SQL Data Definition Language) i iskazi za promene u bazama mogu se izvršiti upotrebom **executeUpdate(String sql)**, a SQL upit se može izvršiti upotrebom **executeQuery(String sql)**. Rezultat se vraća u obliku ResultSet objekta. Na primer, sledeći kod izvršava SQL iskaz **create table Temp (col1 char(5), col2 char(5))**:

```
statement.executeUpdate
("create table Temp (col1 char(5), col2 char(5))");
```

SQL upit:

```
select firstName, mi, lastName
from Student
where lastName = 'Smith' ;
```

se izvršava sledećim kodom:

```
// Select the columns from the Student table
ResultSet resultSet = statement.executeQuery
("select firstName, mi, lastName
from Student
where lastName " + " = 'Smith'");
```

OBRADA REZLTATA RESULTSET

*Klasa **ResultSet** sa metodom **next()** čita red po red kolona koje su dobijene kao rezultat izvršenja SQL upita.*

Klasa **ResultSet** održava tabelu čiji se red čita. Početna pozicija reda je **null**. Možete koristiti metod **next()** da se izvršenje pomeri na sledeći red. Koriste se i različiti "getter" metodi za dobijanje vrednosti iz tekućeg reda. Na primer, sledeći kod prikazuje sve rezultate iz sledećeg SQL iskaza:

```
// Iterate through the result and print the student names
while (resultSet.next())
System.out.println(resultSet.getString(1) + " " +
resultSet.getString(2) + " " + resultSet.getString(3));
```

Metodi **getString(1)**, **getString(2)**, i **getString(3)** donose vrednosti u kolonama za attribute **firstName**, **mi**, i **lastName**. Možete, kao alternativu, koristiti i sledeće iskaze: **getString("firstName")**, **getString("mi")**, i **getString("lastName")** za dobijanje iste vrednosti navedene tri kolone. Prvo izvršenje metoda **next()** postavlja tekući red u drugi red, treći red, i tako dalje, do poslednjeg reda.

Listing klase **SimpleJdbc** je kompletan primer povezivanja baze podataka, izvršenja jednostavnog upita i obrade rezultata upita sa JDBC. Program povezuje lokalnu MySQL bazu podataka i prikazuje studenti čiji je prezime Smith.

Iskaz u liniji 7 instalira JDBC drajver za MySQL, a iskaz u linijama 11-13 povezuje lokalnu MySQL bazu. Program kreira **Statement** objekat (linija 16), izvršava SQL iskaz i vrća **ResultSet** objekat (linije 19-21) i dostavlja rezultat upita iz **ResultSet** objekta (linije 24-26). Poslednji iskaz (linija 29) ztvara vezu i oslobađa resurse.

```
1 import java.sql.*;
2
3 public class SimpleJdbc {
4     public static void main(String[] args)
5         throws SQLException, ClassNotFoundException {
6         // Instalisanje JDBC drajvera
7         Class.forName("com.mysql.jdbc.Driver");
8         System.out.println("Driver loaded");
9     }
```

```
10 // Povezivanje baze podataka
11 Connection connection = DriverManager.getConnection
12     ("jdbc:mysql://localhost/javabook" , "scott", "tiger");
13 System.out.println("Database connected");
14
15 // Kreiranje iskaza
16 Statement statement = connection.createStatement();
17
18 // Izvršenje iskaza
19 ResultSet resultSet = statement.executeQuery
20     ("select firstName, mi, lastName from Student where lastName "
21     + " = 'Smith'");
22
23 // Iteracija kroz rezultat i štampanje imena studenata
24 while (resultSet.next())
25     System.out.println(resultSet.getString(1) + "\t" +
26         resultSet.getString(2) + "\t" + resultSet.getString(3));
27
28 // Zatvaranje veze
29 connection.close();
30 }
31 }
```

VIDEO: UVOD U JDBC

Java JDBC Tutorial - Part 0: Overview (11,24)

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

VIDEO: POVEZIVANJE APLIKACIJE SA MYSQL BAZOM PODATAKA

Java JDBC Tutorial - Part 1: Connect to MySQL database with Java (9,05 minuta)

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

VIDEO: UBACIVANJE PODATAKA U MYSQL BAZU PODATAKA

Java JDBC Tutorial - Part 2: Insert Data into a MySQL Database (4,05)

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

VIDEO: PROMENA PODATAKA U MYSQL BAZI PODATAKA

Java JDBC Tutorial - Part 3: Updating Data in a MySQL Database (3,10 minuta)

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

VIDEO: BRISANJE PODATAKA IZ MYSQL BAZE PODATAKA

Java JDBC Tutorial - Part 4: Deleting Data from a MySQL Database (2,50 minuta)

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

▼ 2.1 Pokazni primeri

KONFIGURACIJA XAMPP-A

Cilj ovog primera je upoznavanje sa MySQL bazama podataka

Preuzmite sa sledećeg sajta XAMPP server koji predstavlja skup Apache, MySQL, FileZilla kao i Tomcat servera:

<https://www.apachefriends.org/index.html>

Kada instalirate XAMPP potrebno je da pokrenete MySQL i Apache. Kako bi pokrenuli Apache potrebno je da Vam Skype i Viber budu ugašeni.

Kada su Apache server i MySQL upaljeni potrebno je da kreirate bazu na linku <http://localhost/phpmyadmin> preko PHPMyAdmin panela.

Kreirajte databazu tako da se zove: korisnici

Ući unutar SQL taba u baze korisnici i izvršiti sledeći upit:

```
create table users (  
  
    id int unsigned auto_increment not null,  
    ime varchar(32) not null,  
    prezime varchar(32) not null,  
    adresa varchar(100),  
    telefon varchar(100),  
    godina int,  
    primary key (id)  
  
);
```

Ovaj upit će kreirati tabelu users koji ima identifikator ID, ime, prezime, adresu, telefon kao i godine.

Ako navedemo not null, onda zabranjujemo da ta kolona ima null vrednosti. Null vrednost nam označava da nešto nije poznato, tj. nije definisano. Opcija auto_increment nam omogućava da baza sama generiše vrednost id polja kada ubacujemo nove korisnike. Primarni ključ označava da je korisnik određen id, što znači i da dva korisnika ne mogu da imaju isti id.

Zašto je bitno da postoji polje koje je jedinstveno, tj. bez duplikata?

PRIMER 16 - KONEKCIJA NA BAZU

Cilj ovog primera je da student nauči kako da koristi konekciju na bazu u Javi

Napraviti program koji upisuje jednog korisnika u bazu sa podacima po izboru

Kako bi se povezali na MySQL preko Jave potreban nam je MySQL connector koji možete preuzeti sa sledećeg sajta:

<http://www.java2s.com/Code/Jar/m/Downloadmysqlconnectorjava5123binjar.htm>

Na dnu strane imate download.

Ubacite biblioteku u novi NetBeans projekat.

Prvo su nam potrebni parametri za konekciju na bazu. Url je putanja do baze, user nam je username za logovanje i imamo password. Ovde koristimo default vrednosti kada smo pravili bazu, korisničko ime je root, a šifra je prazan string. DriverManager.getConnection metod nam služi da se konektujemo na bazu sa opisanim parametrima. createStatement pravi naredbu, a execute je izvršava. Naredba za ubacivanje je INSERT INTO. Navodimo naziv tabele, kolone koje popunjavamo, a zatim i vrednosti koje želimo da unesemo posle reči VALUES. U radu sa bazom može doći do raznih problema, tako da je kod neophodno staviti u try catch blok. Kada završimo sa korišćenjem konekcije obavezno moramo da je zatvorimo sa close metodom.

Klasa Main:

```
public class Main {

    /*
    Kreirati bazu korisnici i u njoj tabelu:
    create table users (
    id int unsigned auto_increment not null,
    ime varchar(32) not null,
    prezime varchar(32) not null,
    adresa varchar(100),
    telefon varchar(100),
    godina int,
    primary key (id)
    );
    */
    /**
    * @param args the command line arguments
    */
    private java.sql.Connection con = null;
    private String url = "jdbc:mysql://localhost/korisnici";
    private String user = "root";
    private String password = "";

    public static void main(String[] args) {

        new Main();
    }

    public Main(){
        try {
            con = DriverManager.getConnection(url, user, password);
            Statement st = (Statement) con.createStatement();
            st.execute("INSERT INTO users (ime,prezime,adresa,telefon,godina) "
                + "" + "VALUES ('Vuk','Vasic','Pere perica
22','06592988888',23)");
            con.close();
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
}
}
```

PRIMER 17 - UPISIVANJE OBJEKATA U BAZU PODATAKA

Cilj ovog primera je da student nauči kako da upisuje klasu direktno u bazu

Napraviti program koji upisuje jednog korisnika u bazu sa podacima po izboru. Korisnik u ovom zadatku treba da bude klasa za sebe koju treba da nazove User:

Klasa User:

```
public class User {
    private String ime;
    private String prezime;
    private String adresa;
    private String telefon;
    private int godine;

    public User() {
    }

    public User(String ime, String prezime, String adresa, String telefon, int
godine) {
        this.ime = ime;
        this.prezime = prezime;
        this.adresa = adresa;
        this.telefon = telefon;
        this.godine = godine;
    }

    public String getIme() {
        return ime;
    }

    public void setIme(String ime) {
        this.ime = ime;
    }

    public String getPrezime() {
        return prezime;
    }

    public void setPrezime(String prezime) {
        this.prezime = prezime;
    }

    public String getAdresa() {
        return adresa;
    }

    public void setAdresa(String adresa) {
        this.adresa = adresa;
    }

    public String getTelefon() {
        return telefon;
    }
}
```

```
public void setTelefon(String telefon) {
    this.telefon = telefon;
}

public int getGodine() {
    return godine;
}

public void setGodine(int godine) {
    this.godine = godine;
}

}
```

Klasa Baza:

```
public class Baza {

    private static java.sql.Connection con = null;
    private static String url = "jdbc:mysql://localhost/korisnici";
    private static String username = "root";
    private static String password = "";

    public static void addUser(User user) {
        try {
            con = DriverManager.getConnection(url, username, password);
            Statement st = (Statement) con.createStatement();
            st.execute("INSERT INTO users (ime, prezime, adresa, telefon, godina) "
                + "" + "VALUES ('" + user.getIme() + "','" + user.getPrezime()
+
                "','" + user.getAdresa() + "','" + user.getTelefon() +
                "','" + user.getGodine() + ")");
            con.close();
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
}
```

Klasa Main:

```
public class Main {
    public static void main(String[] args) {
        new Main();
    }

    public Main(){
        Baza.addUser(new User("Pera", "Peric", "Pere Perica 15", "0647878787", 12));
    }
}
```

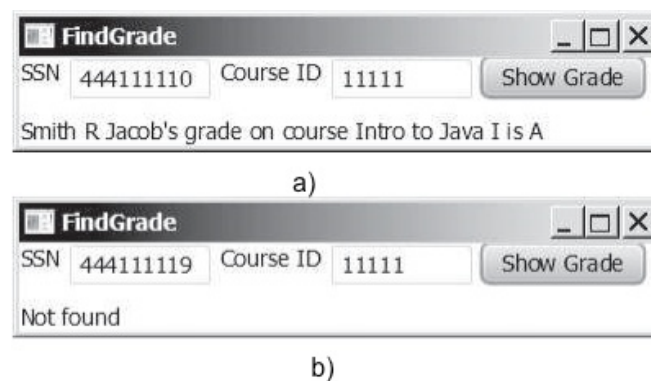
U ovom primeru želimo da objekat neke klase prosledimo metodi koja će da ga upiše u bazu. Takav način programiranja odgovara objektno orijentisanim principima. Imamo klasu User koja čuva podatke o korisniku, zatim klasu Baza koja ima metod addUser i treba da upise prosleđenog korisnika u bazu. Upit je isti, ali sada podatke uzimamo iz objekta pomoću getera i gradimo string.

Svaki put kada dodajemo korisnika mi otvaramo novu konekciju. Zašto e to radi i da li je to dobar način?

PRIMER 18 - PRISTUP BAZI PODATAKA IZ JAVA FX

Moguće je povezati JavaFX klijentski program sa bazom podataka na serveru.

U ovoj sekciji ćemo dati primer povezivanja baze podataka iz JavaFX programa. Program omogućava da korisnik unese SSN (Social Security Number) i ID predmeta da nađu ocenu studenta, kao što je prikazano na slici 5. Listing klase FindGrade upotrebljava MySQL bazu podataka na lokalnom računaru.



Slika 2.2.1 JavaFX klijent može da pristupi bazi podataka na serveru

Metod **initializeDB()** (linije 42–62) instaliraju MySQL drajver (linija 45), povezuje MySQL bazu podataka na računaru liang.armstrong.edu (linije 50–51) i kreira iskaz (linija 57).

Listing klase FindGrade:

```
1 import javafx.application.Application;
2 import javafx.scene.Scene;
3 import javafx.scene.control.Button;
4 import javafx.scene.control.Label;
5 import javafx.scene.control.TextField;
6 import javafx.scene.layout.HBox;
7 import javafx.scene.layout.VBox;
8 import javafx.stage.Stage;
9 import java.sql.*;
10
11 public class FindGrade extends Application {
12     // Iskaz za izvršenje upita
13     private Statement stmt;
```

```

14 private TextField tfSSN = new TextField();
15 private TextField tfCourseId = new TextField();
16 private Label lblStatus = new Label();
17
18 @Override // Redefinisanje metoda start() u klasi Application
19 public void start(Stage primaryStage) {
20 // Inicijalizacija veze baze podataka i kreiranje Statement objekta
21 initializeDB();
22
23 Button btShowGrade = new Button("Show Grade");
24 HBox hbox = new HBox(5);
25 hbox.getChildren().addAll(new Label("SSN"), tfSSN,
26     new Label("Course ID"), tfCourseId, (btShowGrade));
27
28 VBox vbox = new VBox(10);
29 vbox.getChildren().addAll(hbox, lblStatus);
30
31 tfSSN.setPrefColumnCount(6);
32 tfCourseId.setPrefColumnCount(6);
33 btShowGrade.setOnAction(e -> showGrade());
34
35 // Kreiranje scene i njeno postavljanje na pozornicu
36 Scene scene = new Scene(vbox, 420, 80);
37 primaryStage.setTitle("FindGrade"); // Unos naziva pozornice
38 primaryStage.setScene(scene); // Postavljanje scene u pozornicu
39 primaryStage.show(); // prikaz pozornice
40 }
41
42 private void initializeDB() {
43 try {
44 // Instalacija JDBC drajvera
45 Class.forName("com.mysql.jdbc.Driver");
46 // Class.forName("oracle.jdbc.driver.OracleDriver");
47 System.out.println("Driver loaded");
48
49 // Uspostavljanje veze
50 Connection connection = DriverManager.getConnection
51     ("jdbc:mysql://localhost/javabook", "scott", "tiger");
52 // ("jdbc:oracle:thin:@liang.armstrong.edu:1521:orcl",
53 // "scott", "tiger");
54 System.out.println("Database connected");
55
56 // Kreiranje iskaza
57 stmt = connection.createStatement();
58 }
59 catch (Exception ex) {
60     ex.printStackTrace();
61 }
62 }
63
64 private void showGrade() {
65 String ssn = tfSSN.getText();
66 String courseId = tfCourseId.getText();

```

```

67 try {
68     String queryString = "select firstName, mi, " +
69         "lastName, title, grade from Student, Enrollment, Course " +
70         "where Student.ssn = '" + ssn + "' and Enrollment.courseId = '" +
71         " + courseId +
72         "' and Enrollment.courseId = Course.courseId " +
73         " and Enrollment.ssn = Student.ssn";
74
75     ResultSet rset = stmt.executeQuery(queryString);
76
77     if (rset.next()) {
78         String lastName = rset.getString(1);
79         String mi = rset.getString(2);
80         String firstName = rset.getString(3);
81         String title = rset.getString(4);
82         String grade = rset.getString(5);
83
84         // Display result in a label
85         lblStatus.setText(firstName + " " + mi +
86             " " + lastName + "'s grade on course " + title + " is " +
87             grade);
88     } else {
89         lblStatus.setText("Not found");
90     }
91 }
92 catch (SQLException ex) {
93     ex.printStackTrace();
94 }
95 }
96 }

```

PRIMER 19 - ČUVANJE PODATAKA U BAZI

Cilj ovog primera je da student nauči kako da upisuje klasu direktno u bazu sa zaštitom

Napraviti program koji upisuje jednog korisnika u bazu sa podacima po izboru. Korisnik u ovom zadatku treba da bude klasa za sebe koju treba da nazove User. U prethodnom zadatku nismo koristili pripremljeni upit pa je neko mogao da upiše apostrof kao ime korisnika i prekine izvršavanje upita

Klasa User:

```

public class User {
    private String ime;
    private String prezime;
    private String adresa;
    private String telefon;
    private int godine;
}

```



```
public User() {  
}  
  
    public User(String ime, String prezime, String adresa, String telefon, int  
godine) {  
        this.ime = ime;  
        this.prezime = prezime;  
        this.adresa = adresa;  
        this.telefon = telefon;  
        this.godine = godine;  
    }  
  
    public String getIme() {  
        return ime;  
    }  
  
    public void setIme(String ime) {  
        this.ime = ime;  
    }  
  
    public String getPrezime() {  
        return prezime;  
    }  
  
    public void setPrezime(String prezime) {  
        this.prezime = prezime;  
    }  
  
    public String getAdresa() {  
        return adresa;  
    }  
  
    public void setAdresa(String adresa) {  
        this.adresa = adresa;  
    }  
  
    public String getTelefon() {  
        return telefon;  
    }  
  
    public void setTelefon(String telefon) {  
        this.telefon = telefon;  
    }  
  
    public int getGodine() {  
        return godine;  
    }  
}
```

```

    public void setGodine(int godine) {
        this.godine = godine;
    }

}

```

Klasa Baza:

```

public class Baza {

    private static java.sql.Connection con = null;
    private static String url = "jdbc:mysql://localhost/korisnici";
    private static String username = "root";
    private static String password = "";

    public static void addUser(User user) {
        try {
            con = DriverManager.getConnection(url, username, password);
            PreparedStatement st = con.prepareStatement("INSERT INTO users
(ime, prezime, adresa, telefon, godine) " + "VALUES (?, ?, ?, ?, ?)");
            st.setString(1, user.getIme());
            st.setString(2, user.getPrezime());
            st.setString(3, user.getAdresa());
            st.setString(4, user.getTelefon());
            st.setInt(5, user.getGodine());
            st.execute();
            con.close();
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
}

```

Klasa Main:

```

public class Main {
    public static void main(String[] args) {
        new Main();
    }

    public Main(){
        Baza.addUser(new User("Pera", "Peric", "Pere Perica 15", "0647878787", 12));
    }
}

```

Problem kod prethodnog primera jeste što umesto podatka koji želimo da upišemo u bazu neko može da unese string koji će da uradi nešto što ne želimo (sql injection). Podatke unosi korisnik pa moramo biti oprezni. Pored toga, kod u prethodnom primeru je podložan greškama. Koristimo PreparedStatement i prepareStatement metod. Pripremimo šablon upita, ali umesto podataka stavimo upitnike. Posle postavljamo prave vrednosti sa setString ili setInt

u ovom primeru. Na taj način će se proveriti da li su to stvarno podaci ili neke naredbe koje korisnik ne bi smeo da izvršava. Na kraju izvršimo execute da bi se naredba izvršila nad bazom.

Kakvu to naredbu može korisnik da prosledi bazi, a koju mi ne želimo?

PRIMER 20 - PREUZIMANJE PODATAKA IZ BAZE

Cilj ovog primera je da student nauči kako da izvlači podatke iz baze

Napraviti program koji ispisuje sve vrednosti korisnika upisane u bazu podataka. Koristiti MySQL SELECT komandu. Podatke smestiti u Listu i tako ih ispisati.

User.java

```
public class User {
    private String ime;
    private String prezime;
    private String adresa;
    private String telefon;
    private int godine;

    public User() {
    }

    public User(String ime, String prezime, String adresa, String telefon, int
godine) {
        this.ime = ime;
        this.prezime = prezime;
        this.adresa = adresa;
        this.telefon = telefon;
        this.godine = godine;
    }

    public String getIme() {
        return ime;
    }

    public void setIme(String ime) {
        this.ime = ime;
    }

    public String getPrezime() {
        return prezime;
    }

    public void setPrezime(String prezime) {
        this.prezime = prezime;
    }
}
```

```
}

public String getAdresa() {
    return adresa;
}

public void setAdresa(String adresa) {
    this.adresa = adresa;
}

public String getTelefon() {
    return telefon;
}

public void setTelefon(String telefon) {
    this.telefon = telefon;
}

public int getGodine() {
    return godine;
}

public void setGodine(int godine) {
    this.godine = godine;
}

}
```

Klasa Baza:

```
public class Baza {

    private static java.sql.Connection con = null;
    private static String url = "jdbc:mysql://localhost/korisnici";
    private static String username = "root";
    private static String password = "";

    public static List<User> getAllUsers() {
        ArrayList<User> listaUsera = new ArrayList<User>();
        try {
            con = DriverManager.getConnection(url, username, password);
            String query = "SELECT * FROM users";
            Statement st = (Statement) con.createStatement();

            ResultSet rs = st.executeQuery(query);
            while (rs.next()) {
                String ime = rs.getString("ime");
                String prezime = rs.getString("prezime");
                String adresa = rs.getString("adresa");
                String telefon = rs.getString("telefon");
                int godine = rs.getInt("godine");
                listaUsera.add(new User(ime, prezime, adresa, telefon, godine));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return listaUsera;
    }
}
```

```

        }
        st.close();
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
    return listaUsera;
}
}

```

Klasa Main:

```

public class Main {
    public static void main(String[] args) {
        new Main();
    }

    public Main(){
        System.out.println(Baza.getAllUsers());
    }
}

```

Ispisujemo podatke iz baze. Koristimo SELECT naredbu. Hocemo sve podatke o korisnicima, zato stavljamo zvezdicu. Metod executeQuery vraća objekat tipa ResultSet koji predstavlja skup korisnika izvučen iz tabele iz baze. Prolazimo kroz taj skup i dodajemo korisnike u listu. Sa next prelazimo na sledećeg korisnika, sve dok next ne vrati false, kada smo stigli do kraja.

PRIMER 21- FILTRIRANJE PODATAKA

Cilj ovog primera je da student nauči kako da izvlači podatke sa filterom iz databaze

Napraviti program koji ispisuje sve vrednosti korisnika upisane u bazu podataka za korisnike koji su stariji od 18 godina. Koristiti MySQL SELECT komandu. Podatke smestiti u Listu i tako ih ispisati.

User.java

```

public class User {
    private String ime;
    private String prezime;
    private String adresa;
    private String telefon;
    private int godine;

    public User() {
    }

    public User(String ime, String prezime, String adresa, String telefon, int

```

```
godine) {
    this.ime = ime;
    this.prezime = prezime;
    this.adresa = adresa;
    this.telefon = telefon;
    this.godine = godine;
}

public String getIme() {
    return ime;
}

public void setIme(String ime) {
    this.ime = ime;
}

public String getPrezime() {
    return prezime;
}

public void setPrezime(String prezime) {
    this.prezime = prezime;
}

public String getAdresa() {
    return adresa;
}

public void setAdresa(String adresa) {
    this.adresa = adresa;
}

public String getTelefon() {
    return telefon;
}

public void setTelefon(String telefon) {
    this.telefon = telefon;
}

public int getGodine() {
    return godine;
}

public void setGodine(int godine) {
    this.godine = godine;
}
}
```

Klasa Baza:

```
public class Baza {

    private static java.sql.Connection con = null;
    private static String url = "jdbc:mysql://localhost/korisnici";
    private static String username = "root";
    private static String password = "";

    public static List<User> getAllUsers() {
        ArrayList<User> listaUsera = new ArrayList<User>();
        try {
            con = DriverManager.getConnection(url, username, password);
            String query = "SELECT * FROM users WHERE godina > 18";
            Statement st = (Statement) con.createStatement();

            ResultSet rs = st.executeQuery(query);
            while (rs.next()) {
                String ime = rs.getString("ime");
                String prezime = rs.getString("prezime");
                String adresa = rs.getString("adresa");
                String telefon = rs.getString("telefon");
                int godine = rs.getInt("godina");
                listaUsera.add(new User(ime, prezime, adresa, telefon, godine));
            }
            st.close();
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
        return listaUsera;
    }
}
```

Klasa Main:

```
public class Main {
    public static void main(String[] args) {
        new Main();
    }

    public Main(){
        System.out.println(Baza.getAllUsers());
    }
}
```

Jedina razlika ovde je u dodatku

WHERE godina > 18

Na taj način mi kažemo da želimo samo korisnike koji imaju više od 18 godina.

PRIMER 22 - FILTRIRANJE I MANUPULACIJA PODACIMA IZ BAZE

Cilj ovog primera je da student nauči kako da izvlači podatke sa više filtera iz baze

Napraviti program koji ispisuje sve vrednosti korisnika upisane u bazu podataka za korisnike koji imaju ili manje od 18 ili više od 19 godina. Koristiti MySQL SELECT komandu. Podatke smestiti u Listu i tako ih ispisati.

User.java

```
public class User {
    private String ime;
    private String prezime;
    private String adresa;
    private String telefon;
    private int godine;

    public User() {
    }

    public User(String ime, String prezime, String adresa, String telefon, int
godine) {
        this.ime = ime;
        this.prezime = prezime;
        this.adresa = adresa;
        this.telefon = telefon;
        this.godine = godine;
    }

    public String getIme() {
        return ime;
    }

    public void setIme(String ime) {
        this.ime = ime;
    }

    public String getPrezime() {
        return prezime;
    }

    public void setPrezime(String prezime) {
        this.prezime = prezime;
    }

    public String getAdresa() {
```



```

        return adresa;
    }

    public void setAdresa(String adresa) {
        this.adresa = adresa;
    }

    public String getTelefon() {
        return telefon;
    }

    public void setTelefon(String telefon) {
        this.telefon = telefon;
    }

    public int getGodine() {
        return godine;
    }

    public void setGodine(int godine) {
        this.godine = godine;
    }
}

```

Klasa Baza:

```

public class Baza {

    private static java.sql.Connection con = null;
    private static String url = "jdbc:mysql://localhost/korisnici";
    private static String username = "root";
    private static String password = "";

    public static List<User> getGodineIzmedju0samnaestIDvadeset() {
        ArrayList<User> listaUsera = new ArrayList<User>();
        try {
            con = DriverManager.getConnection(url, username, password);
            String query = "SELECT * FROM users WHERE godine > 18 AND godine < 21";
            Statement st = (Statement) con.createStatement();

            ResultSet rs = st.executeQuery(query);
            while (rs.next()) {
                String ime = rs.getString("ime");
                String prezime = rs.getString("prezime");
                String adresa = rs.getString("adresa");
                String telefon = rs.getString("telefon");
                int godine = rs.getInt("godine");
                listaUsera.add(new User(ime, prezime, adresa, telefon, godine));
            }
            st.close();
        } catch (SQLException ex) {

```

```

        ex.printStackTrace();
    }
    return listaUsera;
}

public static List<User> getGodineManjeOd0samnestIliStarije0dDvadeset() {
    ArrayList<User> listaUsera = new ArrayList<User>();
    try {
        con = DriverManager.getConnection(url, username, password);
        String query = "SELECT * FROM users WHERE godine < 18 OR godine > 19";
        Statement st = (Statement) con.createStatement();

        ResultSet rs = st.executeQuery(query);
        while (rs.next()) {
            String ime = rs.getString("ime");
            String prezime = rs.getString("prezime");
            String adresa = rs.getString("adresa");
            String telefon = rs.getString("telefon");
            int godine = rs.getInt("godine");
            listaUsera.add(new User(ime, prezime, adresa, telefon, godine));
        }
        st.close();
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
    return listaUsera;
}
}

```

Klasa Main:

```

public class Main {
    public static void main(String[] args) {
        new Main();
    }

    public Main(){
        System.out.println(Baza.getAllUsers());
    }
}

```

Ovde uvodimo novi filter za korisnike, gde tražimo da imaju ili manje od 18 ili više od 19, tj. da ne smeju da imaju 18 i 19 godina. Koristimo reč OR koje nam označava logičko ili.

2.2 Zadaci za samostalni rad

ZADACI ZA SAMOSTALNI RAD STUDENTA 4-7

Cilj zadatka je provera stečenog znanja studenta

Zadatak 4:

Napraviti tabelu transakcija (novac, staro_stanje, novo_stanje) napraviti funkciju koja upisuje u bazu podataka prosleđeni novac dok novo_stanje prebacuje u staro, a u novo upisuje trenutno stanje povećano za novac koji je prosleđen.

Zadatak 5:

Napraviti igricu jurenja dugmeta po ekranu. Igra treba da traje 60 sekundi a rezultat igre je broj klika dugmeta koje beži. Rezultat igre zajedno sa imenom korisnika upisati u tabelu high_score u bazi.

Zadatak 6:

Napraviti tabelu rezultati unutar neke baze podataka. Napraviti java program koji računa pitagorinu teoremu kroz GUI. Rezultat računanja treba smeštati u tabelu rezultati.

Zadatak 7:

Napraviti tabelu hardver koja ima dva polja (komponenta, naziv). Pronaći ime procesora kao i grafičke karte iz jave i upisati u bazu podataka.

ZADACI ZA SAMOSTALNI RAD STUDENTA 8-9

Cilj zadatka je provežbavanje stečenog znanja studenta

Zadatak 8:

Napraviti tabelu autobusi (ime autobusa, broj autobusa, broj mogućih putnika). Napraviti java formu koja će omogućiti pregled svih autobusa i dodavanje istih.

Zadatak 9:

Napraviti tabelu avionske karte (ime leta, sifra leta, imePutnika, prezimePutnika, brojPasosa). Napraviti java formu koja će omogućiti pregled svih prodatih avionskih karti kao i prodaju istih.

▼ Poglavlje 3

PreparedStatement

ŠTA JE PREPAREDSTATEMENT?

PreparedStatement omogućava kreiranje parametrizovane SQL izjave.

Kada se uspostavi veza sa određenom bazom podataka, ona se koristi za slanje SQL iskaza iz vašeg programa u bazu podataka. Interfejs Statement se koristi za izvršenje statičkih SQL iskaza koji ne sadrže nikakve parametre. Interfejsa PreparedStatement proširuje interfejs Statement koristi se za izvršenje prekompiliranje SQL iskaz sa ili bez parametara. Kako su SQL iskazi prekompilirane, one su efikasne za ponovljena izvršenja.

PreparedStatement objekat je kreiran upotrebom metoda `prepareStatement()` u interfejsu Connection. Na primer, sledeći kod kreira PreparedStatement objekat za jedan SQL insert iskaz.

```
PreparedStatement preparedStatement = connection.prepareStatement  
("insert into Student (firstName, mi, lastName) " +  
"values (?, ?, ?)");
```

Iskaz insert ima tri znaka pitanja za označavanje mesta za parametre koji predstavlja vrednosti za **firstName**, **mi**, i **lastName** u slogu tabele **Student**.

Kao podinterfejs interfejsa Statement, PreparedStatement interfejs nasleđuje sve metode definisane u Statement. On takođe obezbeđuje metode za podešavanje parametara u objektu PreparedStatement. Ovi se metodi koriste za podešavanje vrednosti parametara pre izvršenja iskaza ili procedura. Generalno, seter metodi imaju sledeće ime i signaturu.

```
setX(int parameterIndex, X value);
```

gde je **X** parametar tipa , a **parameterIndex** je indeks tog parametra u iskazu. Indeks ima vrednosti od 1 pa nadalje. Na primer, **setString(int parameterIndex, String value)** postavlja **String** vrednost specifičanom parametru.

Sledeći iskazi prenose parametre "Jack", "A" i "Rzan" na mesta **za firstName, mi, i lastName** u **preparedStatement**:

```
preparedStatement.setString(1, "Jack");  
preparedStatement.setString(2, "A");  
preparedStatement.setString(3, "Ryan");
```

Posle postavljanja parametara, možete izvršiti pripremljeni iskaz **pozivom `executeQuery()`** za SELECT iskaz, i **`executeUpdate()`** za DDL iskaz ili za iskaz promene vrednosti.

VIDEO: PRIMENA PREPAREDSTATEMENT KLASE

Java JDBC Tutorial - Part 5: Prepared Statements (4,50 minuta)

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

▼ 3.1 Pokazni primeri

PRIMER SA PRIMENOM KLASE PREPAREDSTATEMENT

Metod `prepareStatement()` u kome se kreira objekat `PreparedStatement` koji na dinamički način unosi vrednosti svojih parametara.

Metodi **`executeQuery()`** i **`executeUpdate()`** su slični metodima sa istim nazivima kao u `Statement` interfejsu, sem što što nemaju parametre jer SQL iskazi su već definisani u metodi **`prepareStatement ()`** u kome se kreira objekat `PreparedStatement`. Ovde se prikazuje listing klase **`FindGradeUsingPreparedStatement`** koja je modifikovana verzija listinga klase **`FindGrade`** koji datu u objektu učenja JDBC.

Ovaj primer radi isto što i klasa **`FindGrade`**, sem što upotrebljava unapred pripremljen iskaz (**`prepared statement`**) koji dinamički unosi parametre. Kod u ovom primeru je skoro isti kao i kod za klasu **`FindGrade`** dat u objektu učenja JDBC. Novi kod je posebno označen. Unapred pripremljen iskaz je definisan u linijama 56-69 sa **`ssn`** i **`courseld`** kao parametara

Unapred pripremljen SQL iskaz je dobijen u liniji 62. Pre izvršenja SQL upita, stvarne vrednosti za **`ssn`** i **`courseld`** se unosi kao parametri u linijama 73-74- Linija 75 izvršava ova unapred definisan iskaz.

Listing klase `FindGradePreparedStatement`:

```
1 import javafx.application.Application;
2 import javafx.scene.Scene;
3 import javafx.scene.control.Button;
4 import javafx.scene.control.Label;
5 import javafx.scene.control.TextField;
6 import javafx.scene.layout.HBox;
7 import javafx.scene.layout.VBox;
8 import javafx.stage.Stage;
9 import java.sql.*;
10
```

```

11 public class FindGradeUsingPreparedStatement extends Application {
12     // PreparedStatement za izvršenje iskaza
13     private PreparedStatement preparedStatement;
14     private TextField tfSSN = new TextField();
15     private TextField tfCourseId = new TextField();
16     private Label lblStatus = new Label();
17
18     @Override // Redefinisanje metoda start() klase Application
19     public void start(Stage primaryStage) {
20         // Inicijalizacije veze baze podataka i kreiranje Statement objekta.
21         initializeDB();
22
23         Button btShowGrade = new Button("Show Grade");
24         HBox hBox = new HBox(5);
25         hBox.getChildren().addAll(new Label("SSN"), tfSSN,
26             new Label("Course ID"), tfCourseId, (btShowGrade));
27
28         VBox vBox = new VBox(10);
29         vBox.getChildren().addAll(hBox, lblStatus);
30
31         tfSSN.setPrefColumnCount(6);
32         tfCourseId.setPrefColumnCount(6);
33         btShowGrade.setOnAction(e -> showGrade());
34
35         // Kreiranje scene i njeno postavljanje na pozornicu
36         Scene scene = new Scene(vBox, 420, 80);
37         primaryStage.setTitle("FindGrade"); // Set the stage title
38         primaryStage.setScene(scene); // Place the scene in the stage
39         primaryStage.show(); // Display the stage
40     }
41
42     private void initializeDB() {
43     try {
44         // Pozivanje JDBC drajvera
45         Class.forName("com.mysql.jdbc.Driver");
46         // Class.forName("oracle.jdbc.driver.OracleDriver");
47         System.out.println("Driver loaded");
48
49         // Uspostavljanje veze
50         Connection connection = DriverManager.getConnection
51             ("jdbc:mysql://localhost/javabook", "scott", "tiger");
52         // ("jdbc:oracle:thin:@liang.armstrong.edu:1521:orcl",
53         // "scott", "tiger");
54         System.out.println("Database connected");
55
56         String queryString = "select firstName, mi, " +
57             "lastName, title, grade from Student, Enrollment, Course " +
58             "where Student.ssn = ? and Enrollment.courseId = ? " +
59             "and Enrollment.courseId = Course.courseId";
60
61         // Kreiranje iskaza
62         preparedStatement = connection.prepareStatement(queryString);
63     }

```

```
64 catch (Exception ex) {
65     ex.printStackTrace();
66 }
67 }
68
69 private void showGrade() {
70     String ssn = tfSSN.getText();
71     String courseId = tfCourseId.getText();
72     try {
73         preparedStatement.setString(1, ssn);
74         preparedStatement.setString(2, courseId);
75         ResultSet rset = preparedStatement.executeQuery();
76
77         if (rset.next()) {
78             String lastName = rset.getString(1);
79             String mi = rset.getString(2);
80             String firstName = rset.getString(3);
81             String title = rset.getString(4);
82             String grade = rset.getString(5);
83
84             // Display result in a label
85             lblStatus.setText(firstName + " " + mi +
86                 " " + lastName + "'s grade on course " + title + " is " +
87                 grade);
88         } else {
89             lblStatus.setText("Not found");
90         }
91     }
92     catch (SQLException ex) {
93         ex.printStackTrace();
94     }
95 }
96 }
```

▼ 3.2 Zadaci za samostalni rad

ZADACI ZA SAMOSTALNI RAD STUDENTA

Cilj je provežbavanje naučenog gradiva

Zadatak 10:

Sve primere od 17-22 doraditi da koriste PreparedStatement. Koja je prednost korišćenja PreparedStatement-a?

▼ Poglavlje 4

CallableStatement

ŠTA JE CALLABLESTATEMENT?

CallableStatement omogućava vam da izvršite memorisane SQL procedure.

Interfejs CallableStatement je projektovan za izvršenje **memorisanih SQL procedura** (SQL-stored procedures). Procedure mogu daimaju parametre: **IN**, **OUT** ili **INOUT**.

- **IN** parametar prima vrednost datu proceduri pri njenom pozivu.
- **OUT** parametar vraća vrednost posle izvršenja procedure a nema nikakvu vrednost u vreme pozivanja procedure.
- **INOUT** parametar sadrži vrednost datu proceduri pri njenom pozivu, i vraća vrednost po izvršenju procedure.

Na primer, sledeća procedura napisana pomoću Oraklovog jezika PL/SQL ima IN parametar p1., OUT parametar p2 i INOUT parametar p3.

```
create or replace procedure sampleProcedure
(p1 in varchar, p2 out number, p3 in out integer) is
begin
/* radi nešto */
kraj za sampleProcedure;
/
```

Objekat CallableStatement se može da kreira upotrebom metoda **prepareCall(String call)** u Connection interfejsu. Na primer, sledeći kod kreira **CallableStatement cstmt** na **Connection connection** za proceduru **sampleProcedure**.

```
CallableStatement callableStatement = connection.prepareCall(
"{call sampleProcedure(?, ?, ?)}");
```

Deo gornjeg iskaza: **{call sampleProcedure(?, ?, ...)}** se smatra da je to **sintaksa kontrolne SQL sekvence** koja obaveštava drajver da kod između znakova uzvika treba da obrađuje drugačije. Drajver raščlanjuje kontrolnu sekvencu i prevodi je u kod koji baza podataka (tj. DBMS) razume. U ovom primeru **sampleProcedure** je tzv. Oracle-ova procedura. Poziv se prevodi u string:

begin sampleProcedure(?, ?, ?); end

i prenosi se u Oracle bazu podataka na izvršenje. Ove procedure možete nazvati i funkcijama. Sintaksa za kreiranje memorisane SQL procedure za neku funkciju je sledeća:

```
{? = call functionName(?, ?, ...)}
```

PRIMER MEMORISANOG ISKAZA U MYSQL BAZI PODATAKA

Možete koristiti metode `execute()` ili `executeUpdate()` za izvršenje procedure zavisno od tipa SQL iskaza, onda upotrebite geter metode za dobijanje vrednosti iz OUT parametara

CallableStatement nasleđuje PreparedStatement. Pored toga, interfejs CallableStatement obezbeđuje metode za registraciju **OUT** parametara i za dobijanje vrednosti iz **OUT** parametara.

Pre nego što se pozove SQL procedura, potrebno je da upotrebite odgovarajuće seter metode da bi preneli vrednosti u **IN** i **INOUT** parametre, i da upotrebite **registerOutParameter ()** metod za registraciju **OUT** i **INOUT** parametara. Na primer, pre nego što pozovete proceduru **sampleProcedure**, sledeći iskazi prenose vrednosti parametrima p1 (**IN**) i p3 (**INOUT**) i registruju parametre p2 (**OUT**) i p3 (**INOUT**):

```
callableStatement.setString(1, "Dallas"); // Dodeljuje Dallas parametru p1
callableStatement.setLong(3, 1); // Dodeljuje 1 parametru p3
// Regstruje OUT parametre
callableStatement.registerOutParameter(2, java.sql.Types.DOUBLE);
callableStatement.registerOutParameter(3, java.sql.Types.INTEGER);
```

Možete koristiti metode **execute()** ili **executeUpdate()** za izvršenje procedure zavisno od tipa SQL iskaza, onda upotrebite geter metode za dobijanje vrednosti iz OUT parametara. Na primer, sledeći iskazi preuzimaju vrednosti iz parametara p2 i p3.

```
double d = callableStatement.getDouble(2);
int i = callableStatement.getInt(3);
```

Definišimo MySQL funkciju koja vraća broj slogova u tabeli koji zadovoljavaju zahtev za specificirane vrednosti za **firstName** i **lastName** u tabeli **Student**

```
/* Za primer unapred memorisanog iskaza. Upotreba MySQL verzije 5 */
drop function if exists studentFound;
delimiter //
create function studentFound(first varchar(20), last varchar(20))
returns int
begin
declare result int;
select count(*) into result
```

```
from Student
where Student.firstName = first and
      Student.lastName = last;
return result;
end;
//
delimiter ;
/* Postoji prostor između delimiter i ; */
```

PRIMER MEMORISANOG ISKAZA U ORACLE BAZI PODATAKA

*Objekat **CallableStatement** sadrži memorisani iskaz, tj.proceduru.*

Ako imate Oracle bazu podataka, funkcija se definiše na sledeći način:

```
create or replace function studentFound
(first varchar2, last varchar2)
/* Ne imenuje se firstName and lastName. */
return number is
numberOfSelectedRows number := 0;
begin
select count(*) into numberOfSelectedRows
from Student
where Student.firstName = first and
      Student.lastName = last;
return numberOfSelectedRows;
end studentFound;
/
```

Pretpostavimo da je funkcija **studentFound** već kreirana u bazi podataka. Listing klase **TestCallableStatement** daje primer koji testira ovu funkciju koja upotrebljava memorisane funkcije.

Program preuzima MySQL drajver (linija 6), povezuje MySQL bazu podataka (linije 7-9), i kreira memorisani iskaz za izvršenje funkcije studentFound (linije 15-16). Prvi parametar funkcije je vraćena vrednost; drugi i treći parametar odgovaraju imenu i prezimenu studenta. Pre izvršenja memorisanog iskaza, program postavlja ime i prezime (linije 24-25) i registruje OUT parametar (linija 26). Iskaz se izvršava u liniji 27. Povratna vrednost funkcije se dobija u liniji 29.

.

Ako je vrednost veća ili jednaka 1, znači da je pronađen student sa specificiranim imenom i prezimenom (slika 1).

Enter student's first name:
 Enter student's last name:
 Jacob Smith is in the database

Enter student's first name:
 Enter student's last name:
 John Smith is not in the database

Slika 4.1.1 Prikaz rezultata izvršenja primera

```
1 import java.sql.*;
2
3 public class TestCallableStatement {
4     /** Kreira novu formu iz TestTableEditor */
5     public static void main(String[] args) throws Exception {
6         Class.forName("com.mysql.jdbc.Driver");
7         Connection connection = DriverManager.getConnection(
8             "jdbc:mysql://localhost/javabook",
9             "scott", "tiger");
10        // Connection connection = DriverManager.getConnection(
11        // ("jdbc:oracle:thin:@liang.armstrong.edu:1521:orcl",
12        // "scott", "tiger");
13
14        // Kreiranje memorisanog iskaza
15        CallableStatement callableStatement = connection.prepareCall(
16            "{? = call studentFound(?, ?)}");
17
18        java.util.Scanner input = new java.util.Scanner(System.in);
19        System.out.print("Enter student's first name: ");
20        String firstName = input.nextLine();
21        System.out.print("Enter student's last name: ");
22        String lastName = input.nextLine();
23
24        callableStatement.setString(2, firstName);
25        callableStatement.setString(3, lastName);
26        callableStatement.registerOutParameter(1, Types.INTEGER);
27        callableStatement.execute();
28
29        if (callableStatement.getInt(1) >= 1)
30            System.out.println(firstName + " " + lastName +
31                " is in the database");
32        else
33            System.out.println(firstName + " " + lastName +
34                " is not in the database");
35    }
36 }
```

VIDEO: PRIMENA MEMORISANIH PROCEDURA 1

Java JDBC Tutorial - Part 6.1: Calling MySQL Stored Procedures with Java (6,17 minuta)

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

VIDEO: PRIMENA MEMORISANIH PROCEDURA 2

Java JDBC Tutorial - Part 6.2: Calling MySQL Stored Procedures with Java (3,15 minuta)

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

VIDEO: PRIMENA MEMORISANIH PROCEDURA 3

Java JDBC Tutorial - Part 6.3: Calling MySQL Stored Procedures with Java (3,52 minuta)

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

VIDEO: PRIMENA MEMORISANIH PROCEDURA 4

Java JDBC Tutorial - Part 6.4: Calling MySQL Stored Procedures with Java (2,56 minuta)

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

VIDEO: ŠTA SU TRANSAKCIJE?

Java JDBC Tutorial - Part 7: JDBC Transactions with MySQL (5,56 minuta)

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

▼ 4.1 Pokazni primeri

PRIMER 22 - PISANJE PROCEDURA

Cilj ovog primera je provežbavanje zvanja procedura iz Java

Napraviti proceduru u MySQL-u koja treba da primi id a potom vrati ime, prezime kao i adresu korisnika koji ima taj ID.

Kreirati Store procedure pokretanjem komande:

```
DROP PROCEDURE IF EXISTS getKorisnikByID;
```

```
delimiter $$
```

```
CREATE PROCEDURE getKorisnikByID(IN IDU INT, OUT IMEU VARCHAR(30),OUT PREZIMEU  
VARCHAR(200), OUT ADRESAU VARCHAR(200))
```

```
BEGIN
```

```
SELECT ime,prezime,adresa INTO IMEU,PREZIMEU,ADRESAU FROM users WHERE id = IDU;
```

```
END;
```

Kako bi pozvali proceduru u MySQL-u iz Java moramo koristiti CallableStatement

Klasa Main:

```
public class Main {

    /**
     *
     * DROP PROCEDURE IF EXISTS getKorisnikByID;
     * delimiter $$
     * CREATE PROCEDURE getKorisnikByID(IN IDU INT, OUT IMEU VARCHAR(30),OUT PREZIMEU
     * VARCHAR(200), OUT ADRESAU VARCHAR(200))
     * BEGIN
     *     SELECT ime,prezime,adresa INTO IMEU,PREZIMEU,ADRESAU FROM users WHERE id =
     * IDU;
     * END;
     */
    private static java.sql.Connection con = null;
    private static String url = "jdbc:mysql://localhost/korisnici";
    private static String username = "root";
    private static String password = "";

    public static void main(String[] args) {
        new Main();
    }

    public Main() {
        try {
```

```
        con = DriverManager.getConnection(url, username, password);
        String query = "CALL getKorisnikByID(?,?,?,?)";
        CallableStatement st = con.prepareCall(query);
        st.setInt(1, 1);
        st.registerOutParameter(2, java.sql.Types.VARCHAR);
        st.registerOutParameter(3, java.sql.Types.VARCHAR);
        st.registerOutParameter(4, java.sql.Types.VARCHAR);
        st.execute();

        System.out.println(st.getString(2) + " " + st.getString(3) + " " +
st.getString(4));

    } catch (Exception e) {
        e.printStackTrace();
    }
}
}
```

▼ 4.2 Zadaci za samostalni rad

ZADACI ZA SAMOSTALNI RAD STUDENTA

Cilj data dva zadatka za individualnu vežbu je da studenti provežbaju zadatke koji se tiču rada sa bazama iz Jave

Zadatak 11:

Napraviti proceduru koja vraća sve korisnike koji imaju više od 15 godina. Prikazati rad procedure kroz Javu.

Zadatak 12:

Napraviti proceduru koja vraća sve korisnike koji imaju između 20 i 25 godina. Prikazati rad procedure kroz Javu.

▼ Poglavlje 5

Dobijanje meta podatka o bazi podataka

ŠTA SU META PODACI BAZE PODATAKA?

JDBC obezbeđuje interfejs `DatabaseMetaData` za dobijanje informacija o bazi podataka, a interfejs `ResultSetMetaData` za dobijanje informacija u specficiranom `ResultSet`.

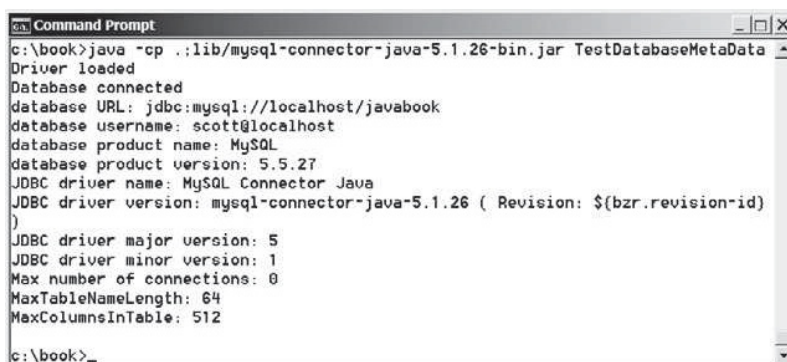
Metapodaci baze podataka, kao što su URL, ime korisnika (username), ime JDBC drajvera, dobijaju se upotrebom `DatabaseMetaData` interfejsa. Skup dobijenih meta podataka, kao što su broj kolona tabele, nazivi kolona, mogu se dobiti upotrebom `ResultSetMetaData` interfejsa.

JDBC obezbeđuje interfejs `DatabaseMetaData` za dobijanje informacija o bazi podataka, a interfejs `ResultSetMetaData` za dobijanje informacija u specficiranom `ResultSet`.

Interfejs `Connection` uspostavlja vezu sa bazom podataka. U kontekstu te veze izvršavaju se SQL iskaze i vraćaju se dobijeni rezultati. Veza takođe obezbeđuje pristup meta podacima baze podataka koji opisuju sposobnosti baze podataka, podržane sa SQL gramatikom, memorisane procedure i dr. Da bi se dobio primerak interfejsa `DatabaseMetaData` za bazu podataka, upotrebite metod **`getMetaData()`** za `Connection` objekat, kao što se ovde prikazuje:

```
DatabaseMetaData dbMetaData = connection.getMetaData();
```

Ako vaš program povezuje MySQL bazu na lokalnom računaru, onda se može primeniti programi, tj. klasa `TestDatabaseMetaData` čiji se listing ovde i koja daje rezultat na slici 1.



```
Command Prompt
c:\book>java -cp .;lib/mysql-connector-java-5.1.26-bin.jar TestDatabaseMetaData
Driver loaded
Database connected
database URL: jdbc:mysql://localhost/javabook
database username: scott@localhost
database product name: MySQL
database product version: 5.5.27
JDBC driver name: MySQL Connector Java
JDBC driver version: mysql-connector-java-5.1.26 ( Revision: ${bcr.revision-id} )
JDBC driver major version: 5
JDBC driver minor version: 1
Max number of connections: 0
MaxTableNameLength: 64
MaxColumnsInTable: 512
c:\book>
```

Slika 5.1.1 Prikaz rezultata izvršenja primera

```
1 import java.sql.*;
2
```

```
3 public class TestDatabaseMetaData {
4     public static void main(String[] args)
5         throws SQLException, ClassNotFoundException {
6         // Pruzimanje JDBC drajvera
7         Class.forName("com.mysql.jdbc.Driver");
8         System.out.println("Driver loaded");
9
10        // azom podataka
11        Connection connection = DriverManager.getConnection
12            ("jdbc:mysql://localhost/javabook", "scott", "tiger");
13        System.out.println("Database connected");
14
15        DatabaseMetaData dbMetaData = connection.getMetaData();
16        System.out.println("database URL: " + dbMetaData.getURL());
17        System.out.println("database username: " +
18            dbMetaData.getUserName());
19        System.out.println("database product name: " +
20            dbMetaData.getDatabaseProductName());
21        System.out.println("database product version: " +
22            dbMetaData.getDatabaseProductVersion());
23        System.out.println("JDBC driver name: " +
24            dbMetaData.getDriverName());
25        System.out.println("JDBC driver version: " +
26            dbMetaData.getDriverVersion());
27        System.out.println("JDBC driver major version: " +
28            dbMetaData.getDriverMajorVersion());
29        System.out.println("JDBC driver minor version: " +
30            dbMetaData.getDriverMinorVersion());
31        System.out.println("Max number of connections: " +
32            dbMetaData.getMaxConnections());
33        System.out.println("MaxTableNameLength: " +
34            dbMetaData.getMaxTableNameLength());
35        System.out.println("MaxColumnsInTable: " +
36            dbMetaData.getMaxColumnsInTable());
37
38        // Zatvaranje veze sa bazom
39        connection.close();
40    }
41 }
```

VIDEO: ŠTA SU META PODACI O BAZI PODATAKA?

Java JDBC Tutorial – Part 8: JDBC Database MetaData with MySQL (5,23 minuta)

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

VIDEO: RESULTSET SA META PODACIMA O BAZI PODATAKA

Java JDBC Tutorial – Part 9: JDBC ResultSet MetaData with MySQL (4,01 minuta)

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

VIDEO: ZAPISIVANJE I ČITANJE BLOB FAJLA?

Java JDBC Tutorial – Part 10: BLOB - Reading and Writing BLOB with MySQL (7,39 minuta)

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

VIDEO: ZAPISIVANJE I ČITANJE CLOB FAJLA

Java JDBC Tutorial – Part 11: CLOB - Reading and Writing CLOB with MySQL (5,52 minuta)

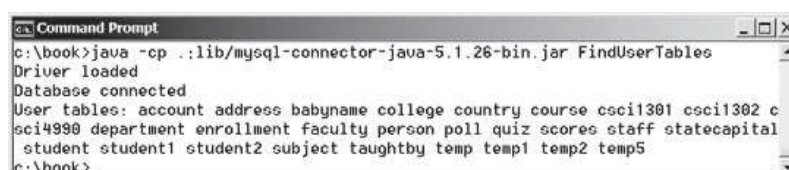
Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ 5.1 Pokazni primeri

PRIMER 23 - DOBIJANJE TABELE BAZE PODATAKA

Metod `getTable()` daje tabele u bazi podataka korisnika.

Možete da identifikujete tabele u bazi podataka preko meta podataka baze upotrebom metoda **`getTable()`**. Listing klase **`FindUserTables`** prikazuje sve tabele korisnika u javabook bazi podataka u lokalnom MySQL bazi podataka. Slika 2 pokazuje dobijeni rezultat izvršenja ovog programa.



```
c:\book>java -cp .:lib/mysql-connector-java-5.1.26-bin.jar FindUserTables
Driver loaded
Database connected
User tables: account address babyname college country course csci1301 csci1302 c
sci4990 department enrollment faculty person poll quiz scores staff statecapital
student student1 student2 subject taughtby temp temp1 temp2 temp5
c:\book>
```

Slika 5.2.1 Prikaz rezultata izvršenja primera

Linija 17 dobija informaciju o tabeli u skupu rezultata upotrebom metoda `getTables()` . Jedna od kolona u skupu rezultata je

Line 17 obtains table information in a result set using the `getTables` method. One of the `TABLE_NAME`. Linija 21 daje nativ tabele iz ovog skupa dobijenih kolona tabele.

Listing klase **FindUserTables**:

```
1 import java.sql.*;
2
3 public class FindUserTables {
4     public static void main(String[] args)
5         throws SQLException, ClassNotFoundException {
6         // Preuzimanje JDBC drajvera
7         Class.forName("com.mysql.jdbc.Driver");
8         System.out.println("Driver loaded");
9
10        // Veza sa bazom podataka
11        Connection connection = DriverManager.getConnection
12            ("jdbc:mysql://localhost/javabook", "scott", "tiger");
13        System.out.println("Database connected");
14
15        DatabaseMetaData dbMetaData = connection.getMetaData();
16
17        ResultSet rsTables = dbMetaData.getTables(null, null, null,
18            new String[] {"TABLE"});
19        System.out.print("User tables: ");
20        while (rsTables.next())
21            System.out.print(rsTables.getString("TABLE_NAME") + " ");
22
23        // Zatvaranje veze
24        connection.close();
25    }
26 }
```

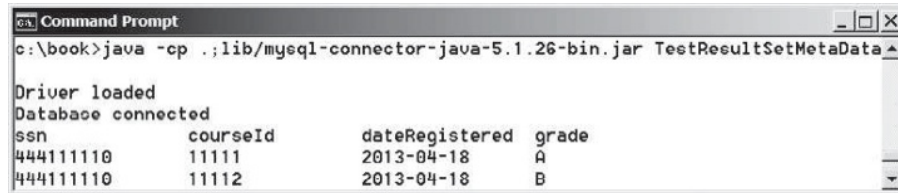
PRIMER 24 - SKUP REZULTATA SA META PODACIMA

Objekat `ResultSetMetaData` se koristi za nalaženje tipova i svojstava kolona u objektu `ResultSet`

Interfejs `ResultSetMetaData` opisuje informaciju koja se odnosi na skup rezultata. Objekat `ResultSetMetaData` se koristi za nalaženje tipova i svojstava kolona u `ResultSet`. Da bi se dobio primerak `ResultSetMetaData`, treba da upotrebite metod `getMetaData()` na skupu rezultata, kao što se ovde pokazuje:

```
ResultSetMetaData rsMetaData = resultSet.getMetaData();
```

Možete da koristite metod **getColumnCount()** za nalaženje broja kolona u rezultatu i metod **getColumnName(int)** za dobijanje naziva kolona. Na primer, listing klase **TestResultSetMetaData** prikazuje sva imena kolona i njihove sadržaje dobijene iz iskaza SQL SELECT iskaza **select * from Enrollment**. Rezultat je prikazan na slici 3.



Slika 5.2.2 Prikaz rezultata izvršenja primera

Listing klase TestResultSetMetaData

```

1 import java.sql.*;
2
3 public class TestResultSetMetaData {
4     public static void main(String[] args)
5         throws SQLException, ClassNotFoundException {
6         // Preuzimanje JDBC drajvera
7         Class.forName("com.mysql.jdbc.Driver");
8         System.out.println("Driver loaded");
9
10        // Veza sa bazom podataka
11        Connection connection = DriverManager.getConnection
12            ("jdbc:mysql://localhost/javabook", "scott", "tiger");
13        System.out.println("Database connected");
14
15        // Kreiranje iskaza
16        Statement statement = connection.createStatement();
17
18        // Izvršenje iskaza
19        ResultSet resultSet = statement.executeQuery
20            ("select * from Enrollment");
21
22        ResultSetMetaData rsMetaData = resultSet.getMetaData();
23        for (int i = 1; i <= rsMetaData.getColumnCount(); i++)
24            System.out.printf("%-12s\t", rsMetaData.getColumnName(i));
25        System.out.println();
26
27        // Prelaz po rezultatu i štampanje imena studenata
28        while (resultSet.next()) {
29            for (int i = 1; i <= rsMetaData.getColumnCount(); i++)
30                System.out.printf("%-12s\t", resultSet.getObject(i));
31            System.out.println();
32        }
33
34        // Zatvaranje veze
35        connection.close();
36    }
  
```

37 }

▼ 5.2 Zadaci za samostalni rad

ZADACI ZA SAMOSTALNI RAD STUDENTA

Cilj zadatka je provežbavanje naučenog

Zadatak 13:

Koristeći meta podatke napraviti program koji vraća i u konzoli prikazuje sve korisnike koji imaju više od 22 godine..

Zadatak 14:

Koristeći meta podatke napraviti program koji vraća i u konzoli prikazuje sve studente koji slušaju predmet sa šifrom KI203.

▼ Poglavlje 6

Razvoj Swing GUI aplikacije sa bazom podataka

VIDEO: POVEZIVANJE JAVA SWING GUI SA MYSQL - UVOD

Java JDBC Tutorial - Part 12.1: Connect Java Swing GUI to a MySQL Database - Overview (2,27 minuta)

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

VIDEO: POVEZIVANJE JAVA SWING GUI SA MYSQL - KREIRANJE DAO

Java JDBC Tutorial - Part 12.2: Connect Java Swing GUI to a MySQL Database - Create the DAO (6,83 minuta)

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

VIDEO: POVEZIVANJE JAVA SWING GUI SA MYSQL - GUI DIZAJN

Java JDBC Tutorial - Part 12.3: Connect Java Swing GUI to a MySQL Database - Design the GUI (3,09 minuta)

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

VIDEO: POVEZIVANJE JAVA SWING GUI SA MYSQL - GUI DOGAĐAJI

Java JDBC Tutorial - Part 12.4: Connect Java Swing GUI to a MySQL Database - GUI Event Handling (3,44 minuta)

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

VIDEO: POVEZIVANJE JAVA SWING GUI SA MYSQL - UNOS PODATAKA U GUI

Java JDBC Tutorial - Part 12.5: Connect Java Swing GUI to a MySQL Database - Populate GUI (4,57 minuta)

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

VIDEO: POVEZIVANJE JAVA SWING GUI SA MYSQL - KREIRANJE FORME ZA UNOS PODATAKA

Java JDBC Tutorial - Part 12.6: Connect Java Swing GUI to a MySQL Database - Create a Form to Add (7,34 minuta)

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

VIDEO: POVEZIVANJE JAVA SWING GUI SA MYSQL - PROMENA PODATAKA U BAZI

Java JDBC Tutorial - Part 12.7: Connect Java Swing GUI to MySQL - Update Database (5,27 minuta)

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

VIDEO: POVEZIVANJE JAVA SWING GUI SA MYSQL - BRISANJE PODATAKA IZ BAZE PODATAKA

Java JDBC Tutorial - Part 12.8: Connect Java Swing GUI to MySQL - Delete an Employee with MySQL (3,14 minuta)

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

VIDEO: POVEZIVANJE JAVA SWING GUI SA MYSQL - KREIRANJE TABELA POVEZIVANJA

Java JDBC Tutorial - Part 12.9: Connect Java Swing GUI to MySQL - Creating Link Tables (6,19 minuta)

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

VIDEO: POVEZIVANJE JAVA SWING GUI SA MYSQL - PROVERA LOZINKE KORISNIKA

Java JDBC Tutorial - Part 12.10: Connect Java Swing GUI to MySQL - Checking User Passwords (7,14 minuta)

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

VIDEO: ČITANJE INFORMACIJA O VEZI SA MYSQL BAZOM IZ FAJLA SA SVOJSTVIMA

Java JDBC Tutorial - Part 13: Reading Database Connection Info from a Properties file with MySQL (3,17 minuta)

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

▼ Poglavlje 7

DOMAĆI ZADATAK

DOMAĆI ZADATAK

Za ove zadatke se ne daje rešenje i očekuje se da svaki student pokuša samostalno rešavanje istih

Zadatak 1:

Za klasu po izboru kreirati JavaFX aplikaciju koja treba da omogući da se svi podaci koji se unose čuvaju u MySQL bazi, da se prikazuju na formi, da mogu da se izmene i obrišu iz baze podataka. Obezbediti i minimum jedno filtriranje po atributu odabrane klase.

Primer:

Napisati klasu Zaposleni koja od atributa ima službeni identifikator - id, ime, prezime, godine starosti i zvanje. Zatim korišćenjem JavaFX kontrola napraviti formu koje će omogućiti unos zaposlenih u MySQL bazu, brisanje, izmenu i prikaz svih podataka iz baze.

▼ Zaključak

REZIME

Pouke ove lekcije.

1. U ovoj lekciji smo predstavili koncept sistema baza podataka, relacionih baza podataka, relacionih modela podataka, integriteta podataka, i SQL. Naučili ste kako da se razvije aplikacija u Javi koja koristi bazu podataka.
2. JAVA API za razvoj aplikacije sa bazama podatak se zove JDBC. JDBC obazbeđuje programerima uniforman interfejs za pristup i za rad sa relacionim bazama podataka.
3. JDBC čine klase i interfejsi za uspostavljanje veza sa bazama podataka, za slanje SQL iskaza bazama podataka (preko DBMS), za obradu rezultata SQL iskaza i za dobijanje meta podataka o bazama.
4. Kako JDBC drajver (uslužni program) služi kao interfejs za olakšavanje komunikacije između JDBC i specifične baze nekog proizvođača (tj. DBMS), JDBC drajveri su specifični za specifične DBMS sisteme. Drajver povezivanja JDBC-ODBC je uključen u JDK radi podržavanja Java programa koji pristupaju bazama podataka preko ODBC drajvera. Ako upotrebljavate drajver koji nije JDBC-ODBC drajver povezivanja (bridž drajver), proverite da li je on upisan u *classpath* pre nego što izvršite program.
5. Pet ključnih interfejsa su potrebna za razvoj bilo koju aplikaciju sa bazama podataka upotrebom Jave: Driver, Connection, Statement, i ResultSet. Ovi interfejsi definišu radni okvir za pristup generičkim SQL bazama podataka. Proizvođači JDBC drajvera obezbeđuju primenu (softver) ovih specifikiranih interfejsa.
6. JDBC aplikacija preuzima odgovarajući drajver upotrebom interfejsa Driver, povezuje se sa bazom podataka upotrebom interfejsa Connection, kreira i izvršava SQL iskaze upotrebom Statement interfejsa i obrađuje dobijen rezultat primenom ResultSet interfejsa.
7. Interfejs PreparedStatement je projektovan za izvršenje dinamičkih SQL iskaza sa parametrima. Ovi SQL iskazi su prekompilirani radi veće efikasnosti kod ponovljenih izvršenja.
8. Meta podaci baze podataka čine informaciju koja opisuje samu bazu podataka. JDBC obezbeđuje DatabaseMetaData interfejs za dobijanje informacije o bazi podataka, a ResultSetMetaData interfejs za dobijanje informacije o određenom ResultSetMetaData objektu.

REFERENCE

Literatura korišćena u ovoj lekciji

1. Y. Daniel Liang, Introduction to Java Programming, Comprehensive Version, Chapter 32, 10th edition, Pierson – preporučeni udžbenik