Administracija baze podataka

Lekcija 5

Dizajn Aplikacije

Sadržaj lekcije

- Razvoj aplikacije & SQL
- Definisanje Transakcija
- Lokovanje
- Batch Obrada

Razvoj db aplikacije i SQL

Da bi dizajnirali aplikaciju, koja se oslanja na bazu za smeštanje podataka, kako treba, dizajner sistema mora minimalno da razume sledeće probleme:

- Kako su podaci smešteni u relacionoj bazi podataka.
- Kako kreirati SQL upite da bi pristupili i modifikovali podatke u bazi podataka
- Kako se SQL razlikuje od tradicionalnih programskih jezika
- Kako embedovati SQL upit u programski jezik aplikacije
- Kako optimizovati pristup bazi podataka izmenom SQL-a i indeksa
- Metode u programiranju za izbegavanje potencijalnih problema u obradi baze podataka

SQL

- SQL je standard za pristupanje relacionim bazama podataka
- SQL je jezik visokog nivoa koji omogućava veći nivo apstrakcije od tradicionalnih proceduralnih jezika
- SQL je dizajniran tako da programer definiše *koji* podaci mu trebaju
  - On ne definiše *kako* se dohvataju

SQL: English-like

- SQL može da se koristi za dohvatanje podataka koristeći English-like sintaksu.
- Lakše je razumeti sledeći kod:

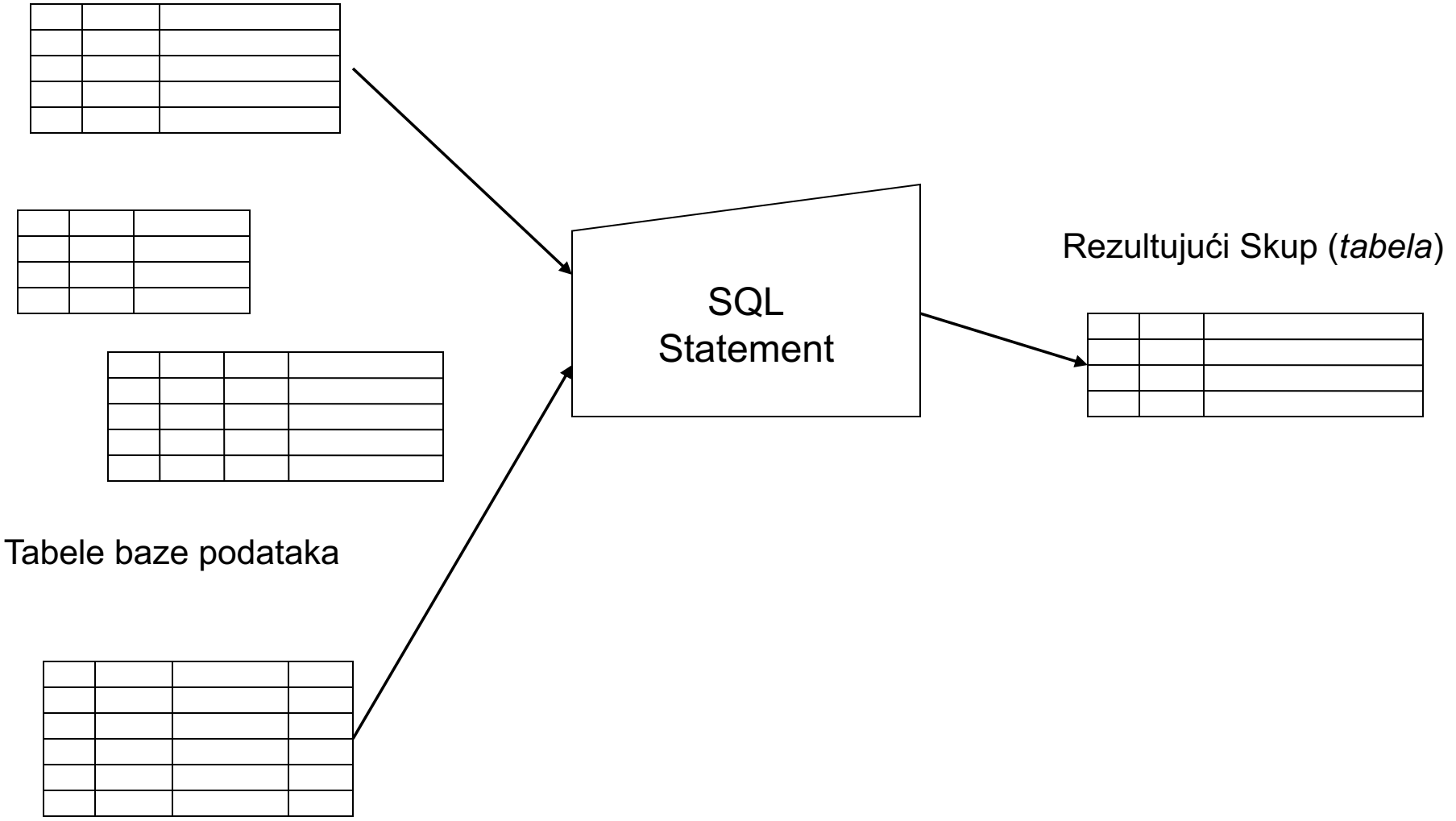
```
SELECT    deptnum, deptname
FROM      dept
WHERE     supervisornum = '903';
```

- Nego razumeti C, Java, ili druge tipične programske jezike

SQL obrada nad skupovima

- SQL funkcioniše nad skupovima
- Više redova može da se dohvati, modifikuje ili ukloni u jednom koraku koristeći jednu SQL naredbu
- Svaka operacija koja se izvodi nad relacionom bazom operiše nad tabelom (ili skupom tabela) i rezultat je druga tabela
- Ovo se naziva *relaciono zatvaranje*

Relaciono Zatvaranje



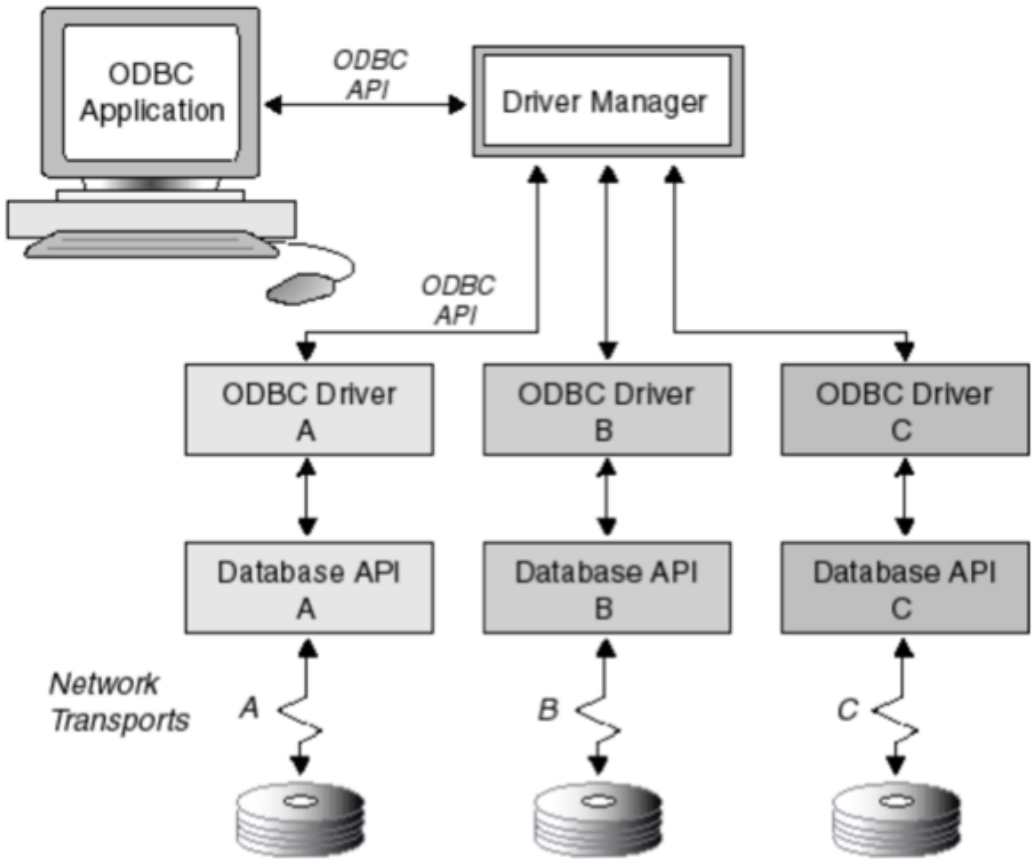
Embedovanje SQL-a u program

- Glavni jezik
  - COBOL, FORTRAN, Assembler, etc.
  - C/C++, Java, PHP, Visual Basic, etc.
- API
  - ODBC, JDBC
- IDE
  - Integrated Development Environment
- Generatori koda

ODBC

- Open Database Connectivity (ODBC) daje standardni interfejs za pristup aplikacije različitim izvorima podataka
- Aplikacija se ne rekompajlira za svaki izvor podataka
- Driver baze podataka povezuje aplikaciju sa određenim izvorom podataka

Komponente ODBC Modela



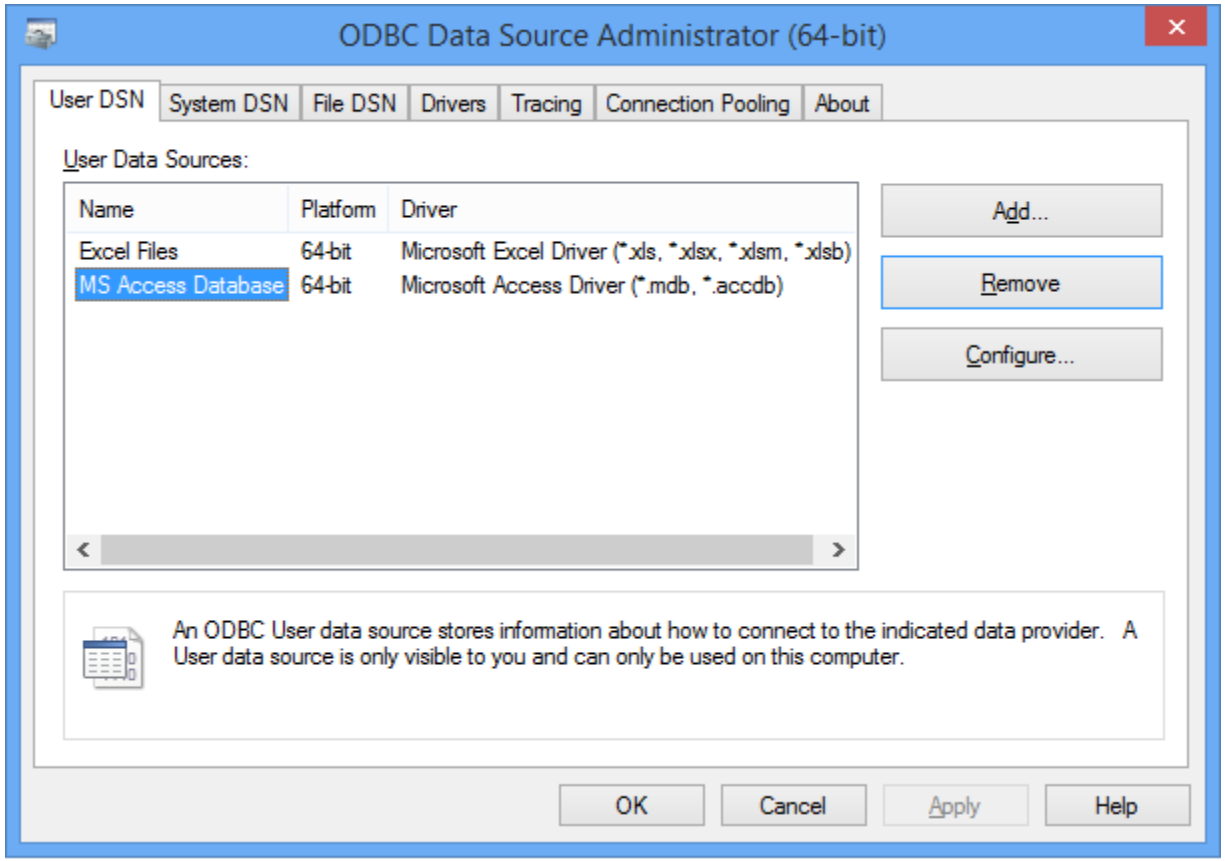
Izvor: docs.oracle.com

ODBC Model

- ODBC aplikacija poziva driver managera (windows, unix) kroz ODBC API
- Driver Manager poziva ODBC driver
- ODBC driver pristupa bazi preko mreže koristeći Database API

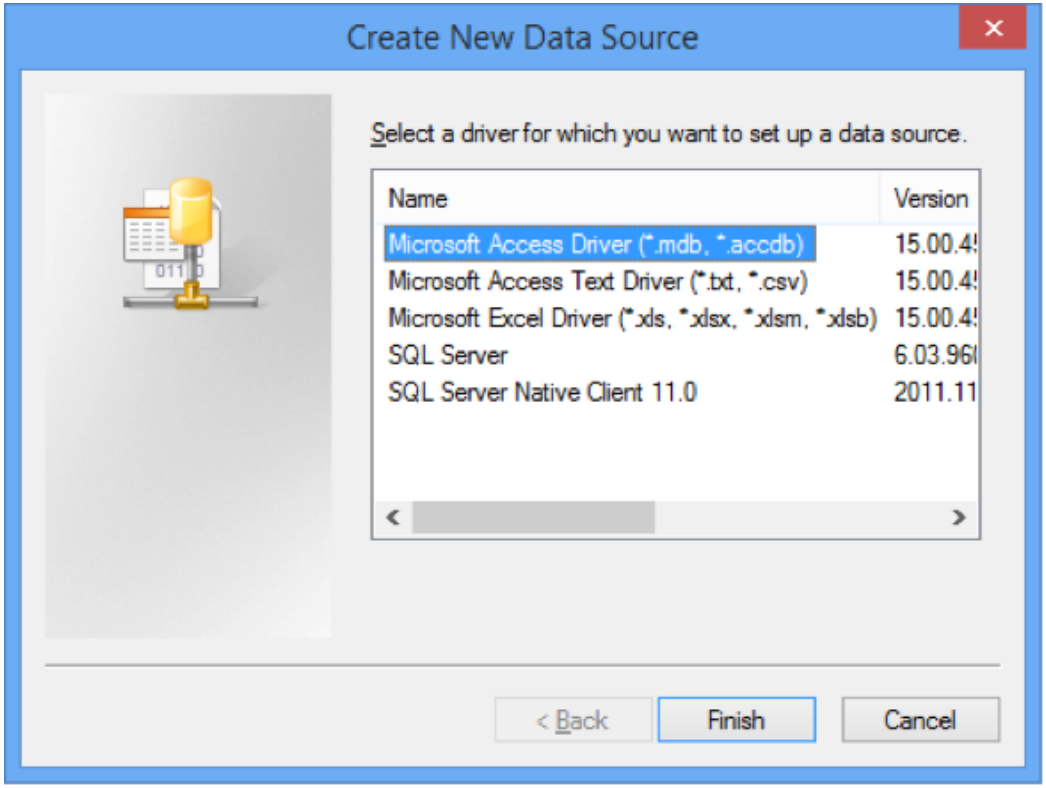
Set up ODBC imena na Windows-u

- ControlPanel->System and Security->Administrative Tools-Data Sources (ODBC) pa Add

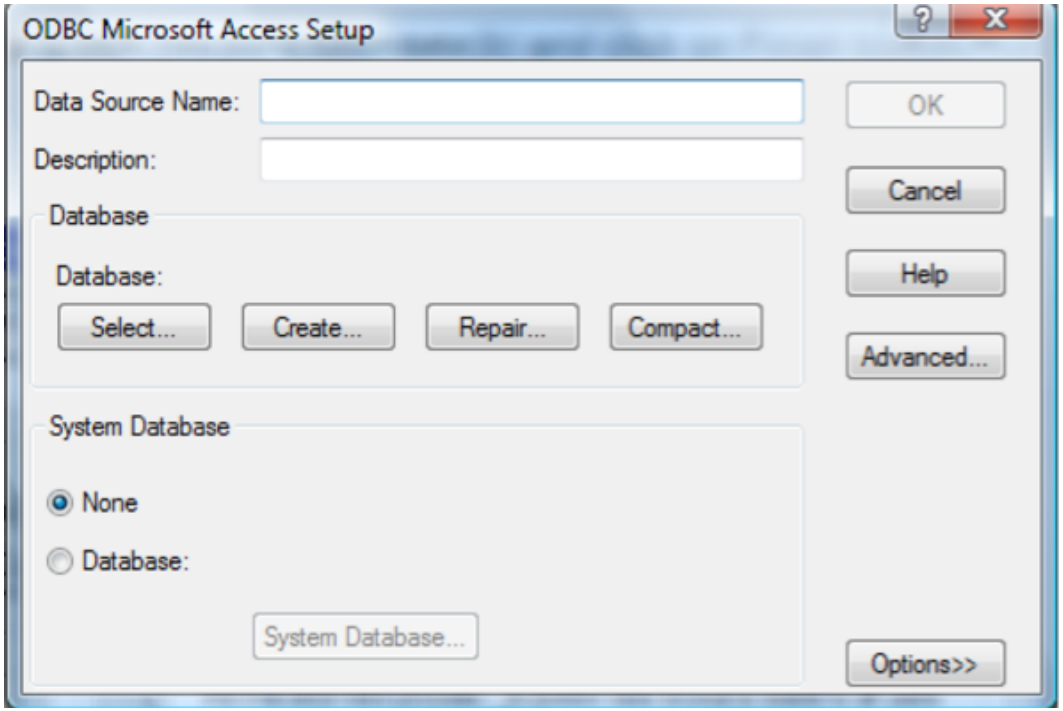


Set up ODBC imena na Windows-u

- Odabrati odgovarajući driver iz liste, koji zavisi od izvora podataka

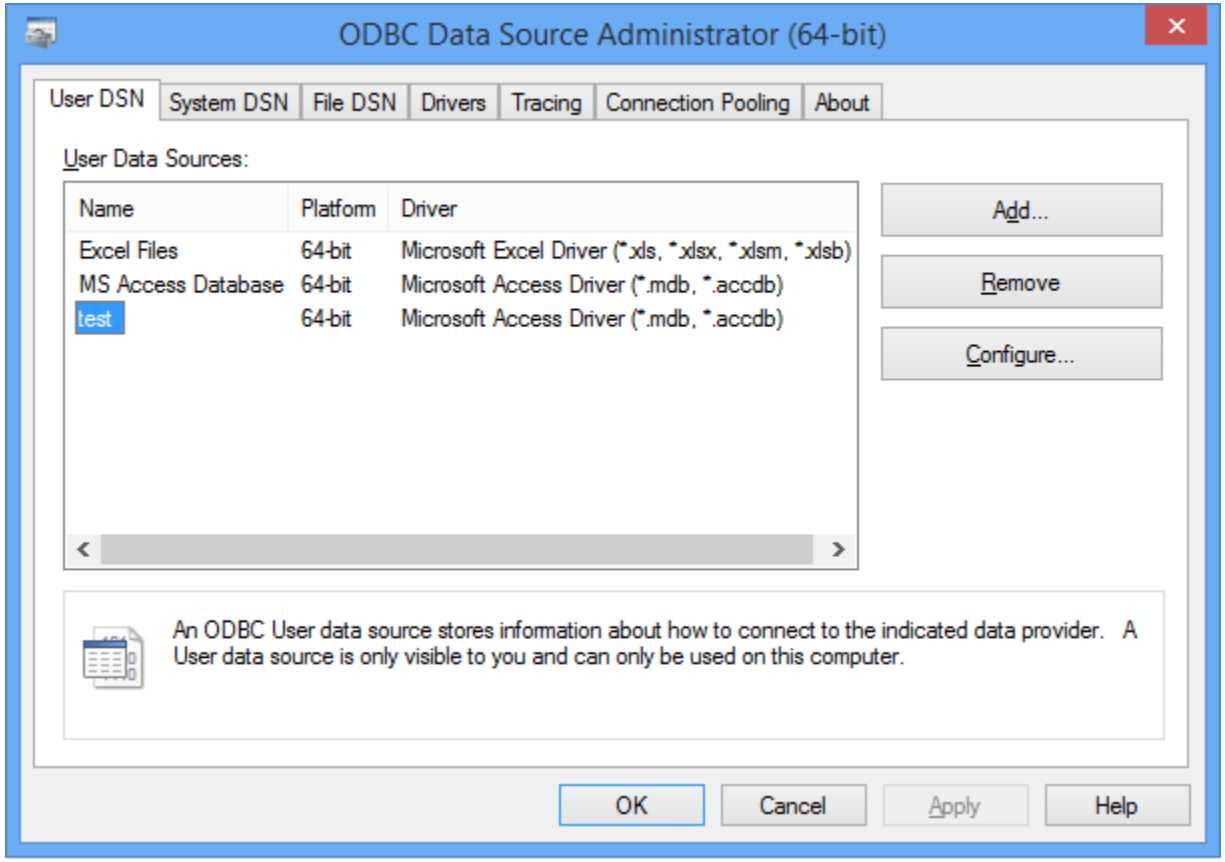


Set up ODBC imena na Windows-u



Set up ODBC imena na Windows-u

- Odabrati ime izvora podataka (na primer test)

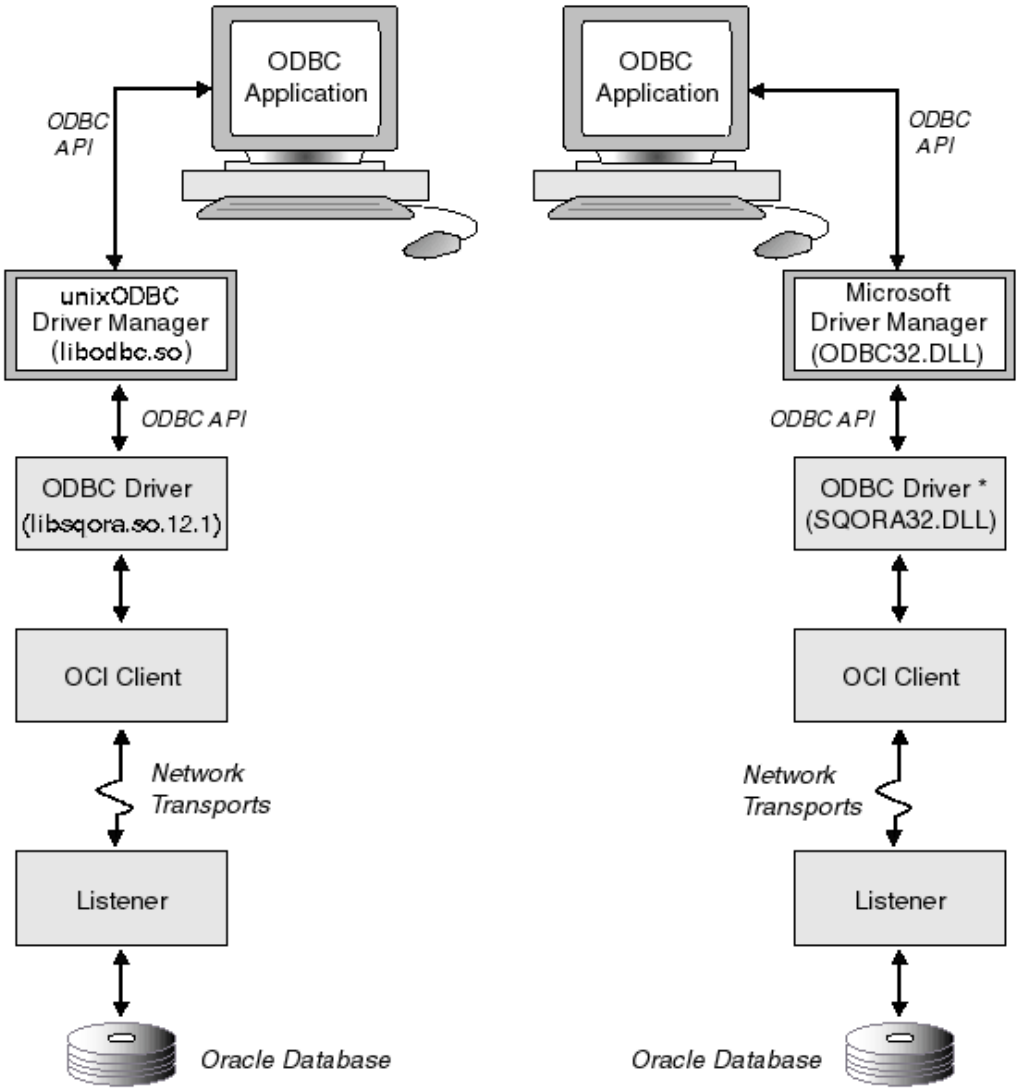


Connection String

C#

```
using System.Data.Odbc;
varconn = newOdbcConnection();
conn.ConnectionString=
                "Dsn=test;"+
                "Uid=UserName;"+
                "Pwd=Secret;";
conn.Open();
```

Oracle ODBC driver



SQL Middleware i API

- JDBC
  - JDBC omogućava Javi pristup ka relacionim bazama podataka.
  - Slično ODBC-u, JDBC se sastoji iz skupa klasa i interfejsa koji mogu da se koriste za pristup relacionim podacima
  - Postoji nekoliko tipova JDBC middleware, uključujući JDBC-do-ODBC bridge, kao i direktna JDBC konekcija ka relacionoj bazi
  - Bilo ko upoznat sa aplikacionim programiranjem i ODBC-om (ili bilo kojim call-level interfejsom) može da pokrene rad sa JDBC-om

Drajveri

- ODBC i JDBC se oslanjaju na *drajvere*
  - Drajver pruža optimizovan interfejs za određenu DBMS implementaciju
- Programi mogu da koriste drajvere da bi komunicirali sa bilo kojom JDBC- or ODBC-kompatibilnom bazom podataka.
- Drajveri omogućavaju standardni skup SQL naredbi u bilo kojoj Windows aplikaciji da bi se transliralo u komande koje su prepoznate od udaljene SQL-kompatibilne baze podataka

JDBC

- JDBC (Java Database Connectivity) zahteva drajvere za svaku individualnu bazu
- JDBC driver je softver koji omogućava Java aplikaciji da komunicira sa bazom
- Java driver omogućava konekciju ka bazi i implementira protokol za transfer upita i rezultata između klijenta i baze

Kategorije JDBC Drivera

1. JDBC-ODBC bridge
2. Native-API driver
3. Network-Protocol driver (Middleware)
4. Database-Protocol driver (pure Java driver) ili thin driver

JDBC-ODBC bridge driver

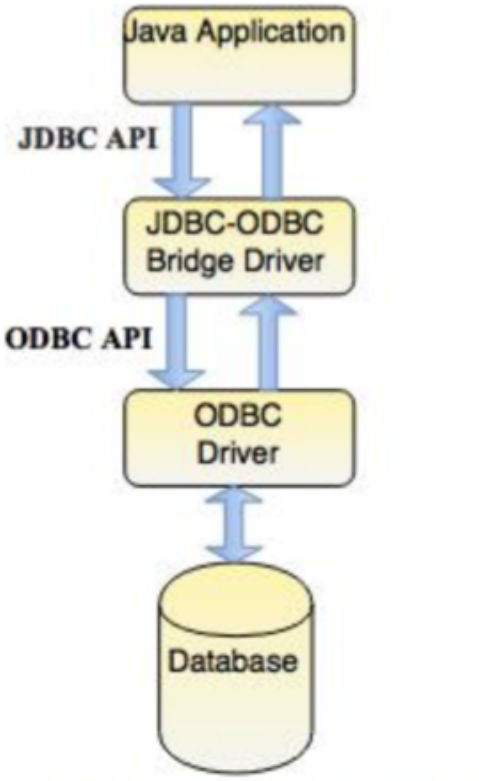


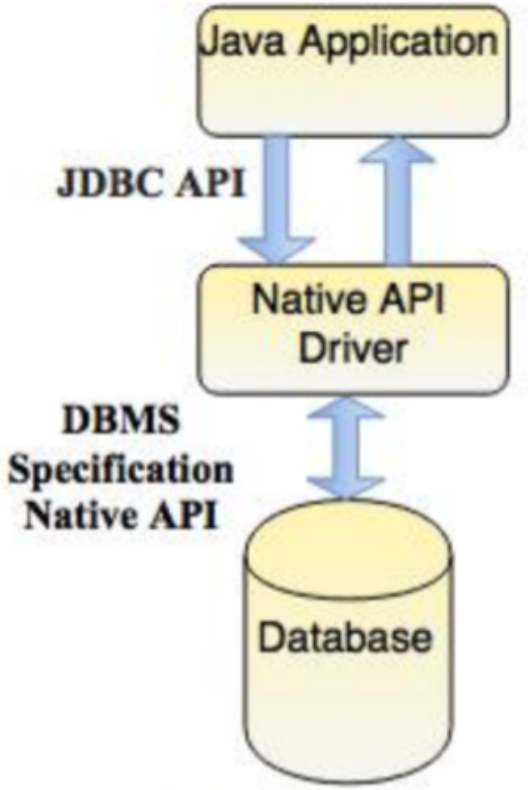
Figure: JDBC-ODBC Bridge Driver

- Koristi ODBC driver da se konektuje na bazu
- Konvertuje JDBC pozive u pozive ODBC-a
- Koristi se kad nema druge alternative
- Izbačen je iz verzije Java 8 (ranije sun.jdbc.odbc.JdbcOdbcDriver)
- Prednost
  - Lako za korišćenje
  - Može da se konektuje na bilo koju bazu
- Mane
  - Performanse degradirane zbog konverzije poziva
  - ODBC driver mora biti instaliran na klijentskoj masini
  - Zahteva kreiranje DSN (Data Source Name) koji predstavlja ciljnu bazu podataka
  - Preporuka je da se koriste JDBC driveri koje je proizvođač baze ponudio

JDBC-ODBC bridge driver

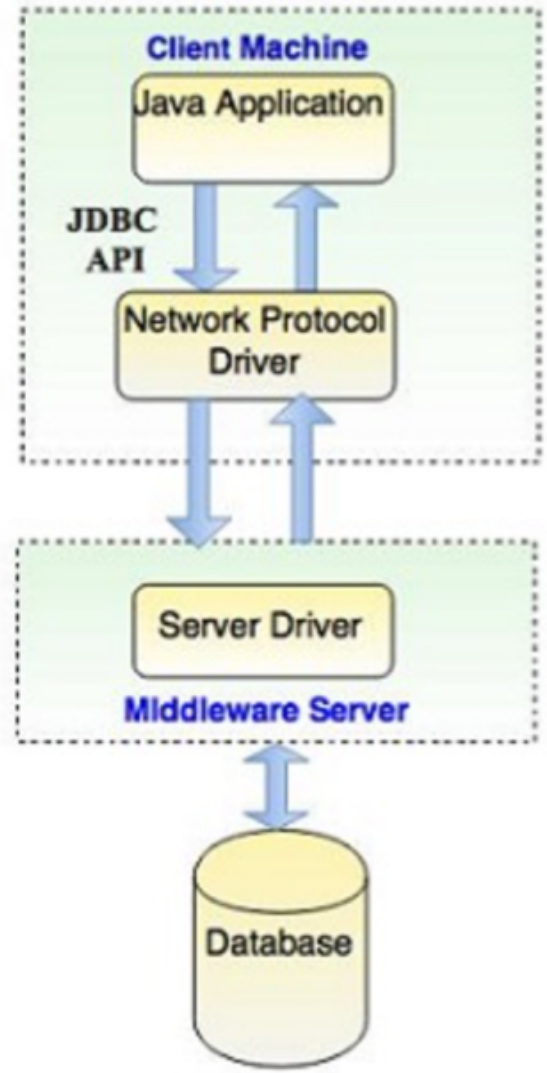
```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.Statement;

public class Main {
  public static void main(String[] args) throws Exception {
    Connection conn = getConnection();
    Statement st = conn.createStatement();
    // st.executeUpdate("drop table survey;");
    st.executeUpdate("create table survey (id int,name varchar(30));");
    st.executeUpdate("insert into survey (id,name ) values (1,'nameValue')");

    st = conn.createStatement();
    ResultSet rs = st.executeQuery("SELECT * FROM survey");

    ResultSetMetaData rsMetaData = rs.getMetaData();

    int numberOfColumns = rsMetaData.getColumnCount();
    System.out.println("resultSet MetaData column Count=" + numberOfColumns);

    st.close();
    conn.close();
  }

  private static Connection getConnection() throws Exception {
    String driver = "sun.jdbc.odbc.JdbcOdbcDriver";
    String url = "jdbc:odbc:northwind";
    String username = "";
    String password = "";
    Class.forName(driver);
    return DriverManager.getConnection(url, username, password);
  }
}
```

Native driver



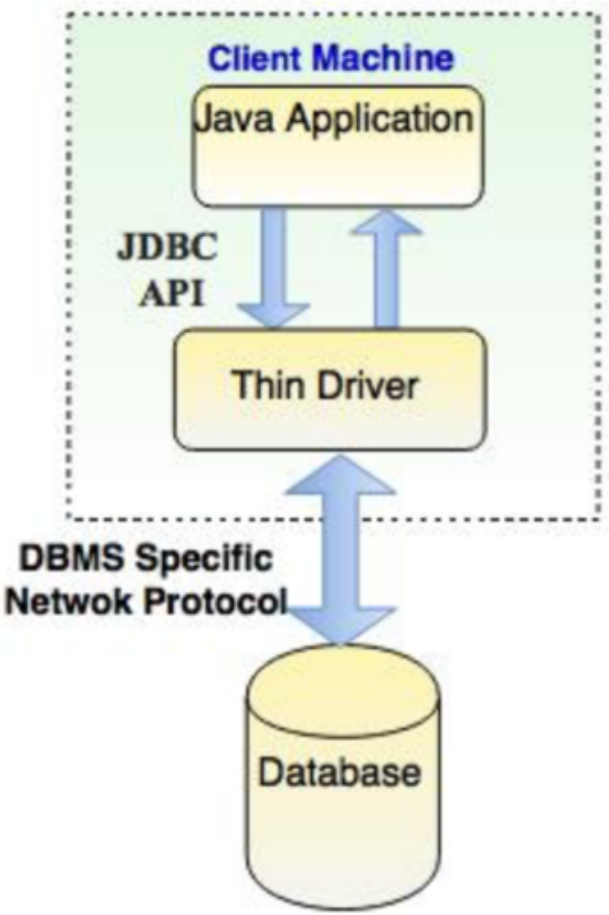
- Koristi biblioteke baze na klijentskoj strani
- Driver kovertuje JDBC metode u native pozive APIja baze
- Nije ceo pisan u Javi
- Prednosti
  - Performanse bolje nego JDBC-ODBC driver
- Mane
  - Native driver mora biti instaliran na klijentskoj masini
  - Ako se promeni baza, mora da se promeni i native driver
- Oracle Call Interface (OCI) driver je primer

JDBC-Net



- Koristi middleware (aplikacioni server) koji konvertuje JDBC pozive u protokol baze
- Pisan ceo u Javi
- Prednosti
  - Nije potrebna biblioteka na strani klijenta
  - Fleksibilnost
- Mane
  - Podrška za mrežu je potrebna na klijentskoj mašini
  - Održavanje drivera košta pošto je potrebno specifično programiranje u srednjem sloju

Thin driver



- Konvertuje JDBC pozive direktno u protokol specifične baze
- Ceo je pisan u Javi
- Komunicira direktno sa bazom kroz socket konekciju
- Prednosti
  - Bolje performanse od svih ostalih drivera
  - Nije potreban softver ni na klijentu ni na serveru
- Mane
  - Driver zavisi od baze
- MySQL's Connector/J driver je primer ove vrste drivera

Thin driver

```
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.SQLException;

public class OracleJDBCExample {

    public static void main(String[] argv) {

        System.out.println("-------- Oracle JDBC Connection Testing ------");

        try {

            Class.forName("oracle.jdbc.driver.OracleDriver");

        } catch (ClassNotFoundException e) {

            System.out.println("Where is your Oracle JDBC Driver?");
            e.printStackTrace();
            return;

        }

        System.out.println("Oracle JDBC Driver Registered!");

        Connection connection = null;

        try {

            connection = DriverManager.getConnection(
                    "jdbc:oracle:thin:@localhost:1521:xe", "system", "password");

        } catch (SQLException e) {

            System.out.println("Connection Failed! Check output console");
            e.printStackTrace();
            return;
```

Konekcija sa ORACLE bazom

```
import java.sql.*;
class OracleCon{
public static void main(String args[]){
try{
  //step1 load the driver class
  Class.forName("oracle.jdbc.driver.OracleDriver");
  //step2 create  the connection object
    Connection con=DriverManager.getConnection(
     "jdbc:oracle:thin:@localhost:1521:xe","system","oracle");
  //step3 create the statement object
  Statement stmt=con.createStatement();
  //step4 execute query
  ResultSet rs=stmt.executeQuery("select * from emp");
  while(rs.next())
   System.out.println(rs.getInt(1)+"  "+rs.getString(2)+"  "+rs.ge
   tString(3));
  //step5 close the connection object
  con.close();
}catch(Exception e){ System.out.println(e);}
}  }
// ojdbc14.jar mora da bude ucitan
```

Konekcija sa Mysql bazom

```
import java.sql.*;
class MysqlCon{
 public static void main(String args[]){
   try{
      Class.forName("com.mysql.jdbc.Driver");
      Connection con=DriverManager.getConnection(

   "jdbc:mysql://localhost:3306/sonoo","root","root");

      Statement stmt=con.createStatement();

   ResultSet rs=stmt.executeQuery("select * from emp");
      while(rs.next())
        System.out.println(rs.getInt(1)+"  "+rs.getString(2
   )+"  "+rs.getString(3));
      con.close();
   }catch(Exception e){ System.out.println(e);}
 }
}

//mysqlconnector.jar mora da bude ucitan
```

Konekcija sa Access bazom

```
import java.sql.*;
class Test{
 public static void main(String args[]){
  try{
   String url="jdbc:odbc:mydsn";  //mydsn je DSN ime
   Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
   Connection c=DriverManager.getConnection(url);
   Statement st=c.createStatement();
   ResultSet rs=st.executeQuery("select * from login
   ");

   while(rs.next()){
    System.out.println(rs.getString(1));
   }
  }catch(Exception ee){System.out.println(ee);}
 }
}
```

Definisanje Transakcija

- Transakcija je atomična jedinica rada u odnosu na oporavak i konzistenciju
- Kada se svi koraci koji čine određenu transakciju odrade, onda se izdaje COMMIT
  - ROLLBACK pre COMMIT za undo onoga što je transakcija uradila
- DBMS održava log transakcije

ACID osobine transakcije

- Definisanje transakcije
  - Atomicity
  - Consistency
  - Isolation
  - Durability
- Jedinica rada
  - Osigurati ispravnu definiciju i kodiranje

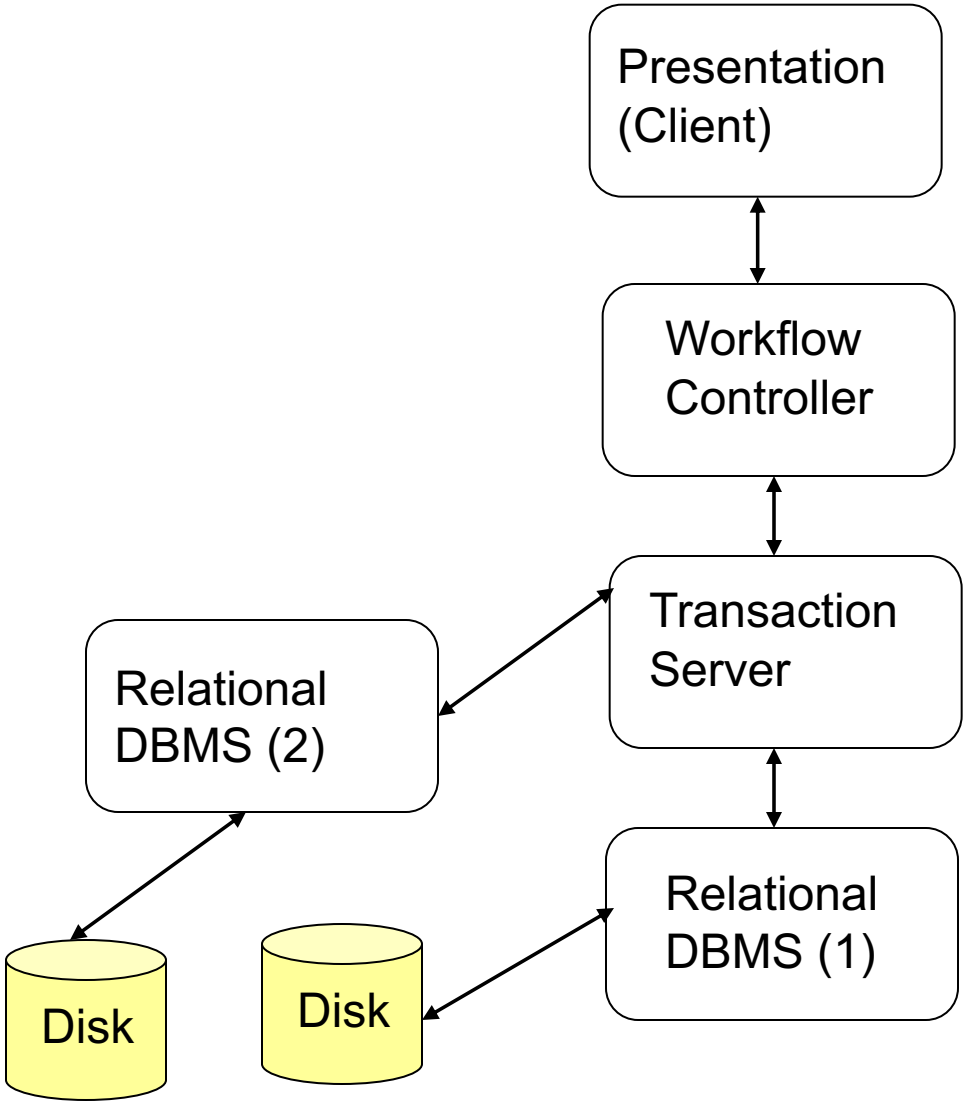
Jedinica posla

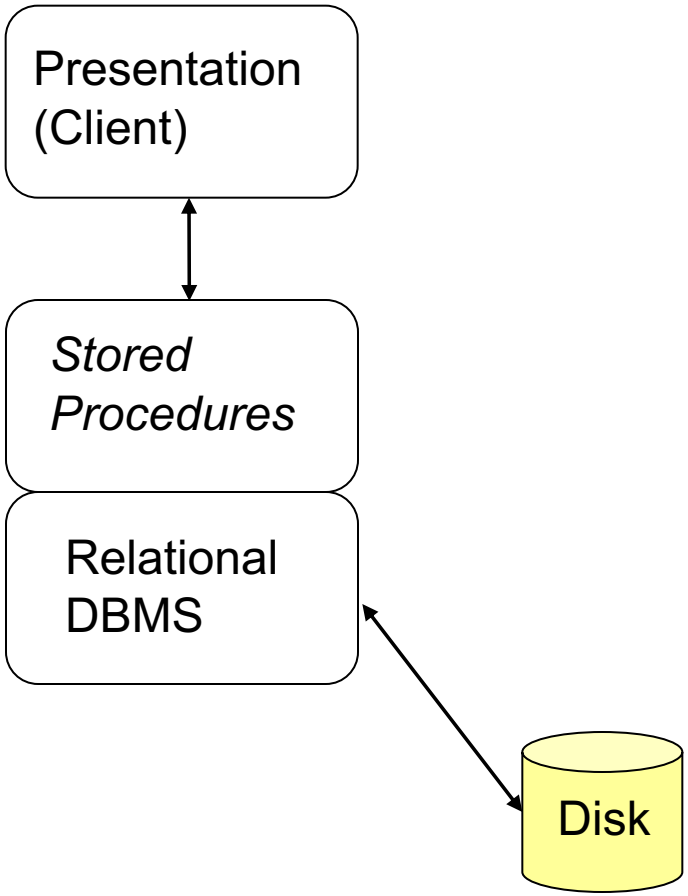
- Jedinica posla je serija instrukcija i poruka koje garantuju integritet podataka.
- Primer: transakcija banke
  - Podizanje $20
  - Transakcija mora da uključi i oduzimanje $20 sa računa i izdavanje $20
  - Nezavisno obavljanje bilo koje od njih nije kompletna jedinica posla

Transaction processing (TP) system

- Podržava obradu transakcija
- Koristan za kritične aplikacije koje zahtevaju veliku količinu konkurentnih korisnika sa minimalnim vremenom zastoja
- Efikasno kontroliše konkurentno izvršavanje aplikacije koji podržavaju veliki broj online korisnika
- Primeri TP sistema, softver: CICS, IMS/TM, Tuxedo, and Microsoft Transaction Server.

TP (transaction processing) Vs DBMS (Stored Procs)





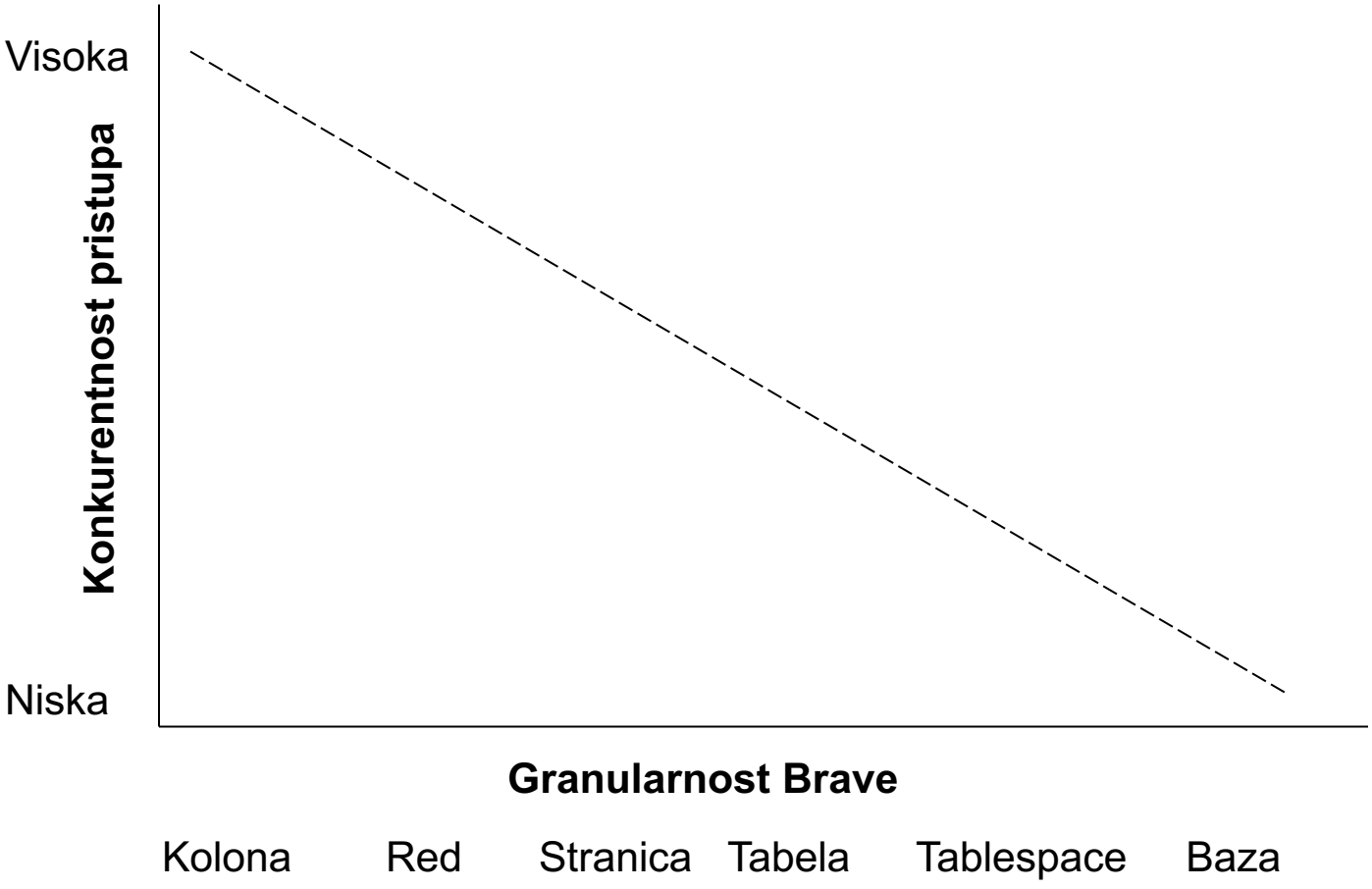
Aplikacioni Serveri

- Aplikacioni server kombinuje osobine servera transakcija zajedno sa dodatnim funkcionalnostima da bi pomogao u kreiranju, održavanju i distribuiranju db aplikacija
- Primeri:
  - WebSphere (IBM)
  - Zend Server (for PHP based apps)
  - Base4 Application Server (open source)

Transakcije i Brave

- DBMS koristi mehanizam *brava* da bi omogućio da više konkurentnih korisnika pristupi i modifikuje podatke u bazi podataka
- Korišćenjem brava, DBMS automatski garantuje integritet podataka.
- DBMS strategije brava omogućavaju da više korisnika iz više okruženja pristupe i modifikuju podatke u bazi podataka u isto vreme.
- Granularnost Brava
  - Red
  - Stranica (ili Blok)
  - Tabela
  - Table Space
  - Baza Podataka

Nivo Granularnosti Brava



Vrste Brava

- Sledeći tipovi brava mogu da se koriste nad stranama baze ili redovima:
  - Deljena brava
    - Uzima se kada se podatak čita bez namere da se ažurira.
    - Ukoliko se deljeno zaključavanje izvrši na redu, strani ili tabeli, ostalim procesima ili korisnicima se dozvoljava čitanje istih podataka
  - Ekskluzivna brava
    - Uzima se kada se podatak modifikuje.
    - Ukoliko se eksluzivna brava uzme na redu, strani ili tabeli, ostalim procesima ili korisnicima se ne dozvoljava čitanje ili izmena istih podataka
  - Update brava
    - Uzima se kada se podatak prvo čita pre nego što se promeni ili obriše.
    - Update brava ukazuje da podatak može da se modifikuje ili briše u budućnosti.
    - Ukoliko se podaci modifikuju ili brišu, DBMS će unaprediti update bravu na ekskluzivnu bravu.

Intent Brava

- Intent brave su smeštene na višem nivou objekata baze podataka kada korisnik ili proces uzme bravu nad stranama podataka ili redovima
  - Tabela ili Prostor Tabele
- Intent brave ostaju sve dok postoje lokovi nižeg nivoa.

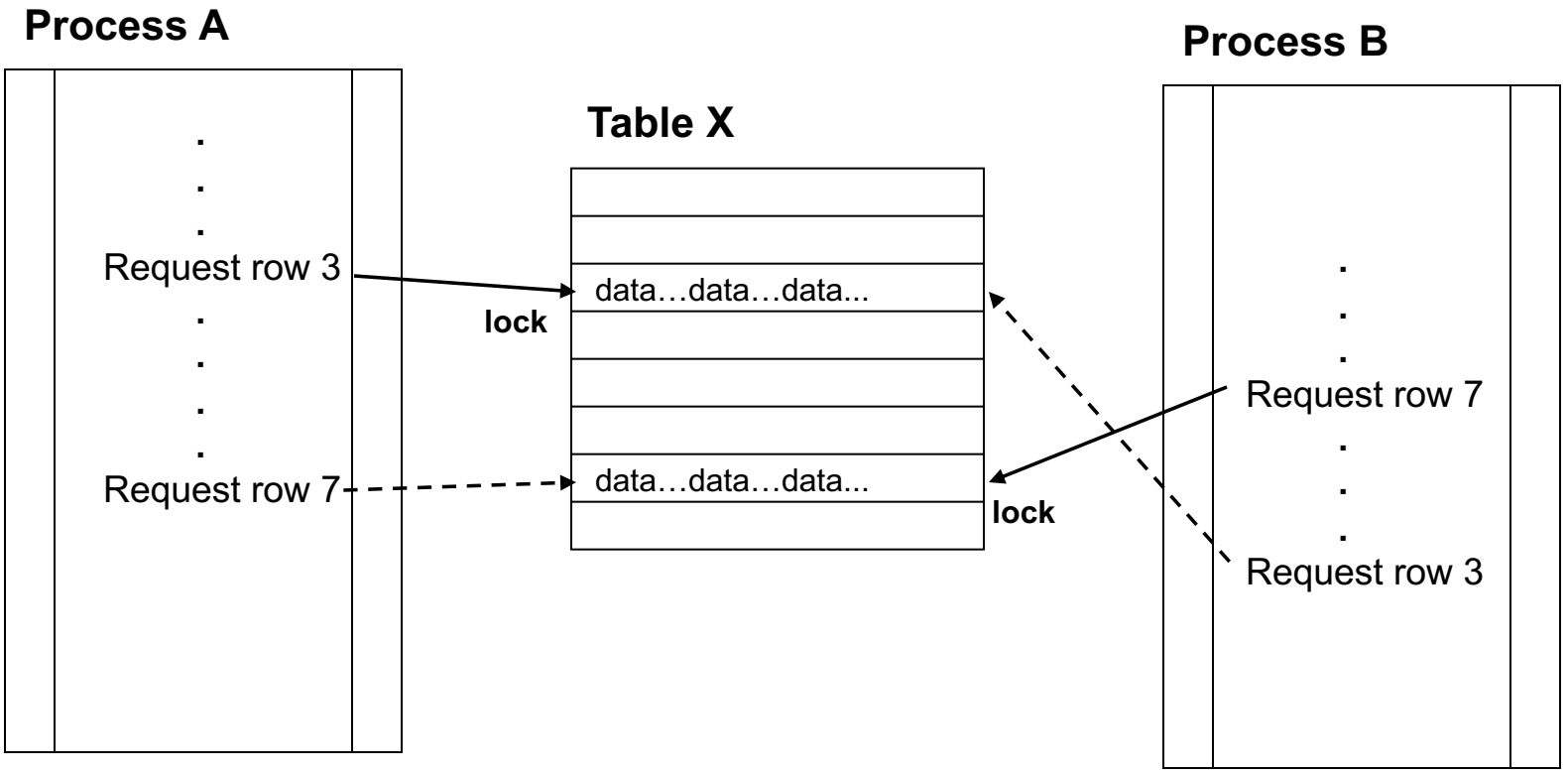
Kompatibilnost Brava

Locks for PGM2	Locks for PGM1		
	S	U	X
S	Yes	Yes	No
U	Yes	No	No
X	No	No	No

Timeout Brave

Program 1		Program 2
Update Table A/Page 1		
Lock established		
Intermediate processing		Update Table A/Page 1
.		Lock (wait)
.		Lock suspension
.	***Timeout***	**error**

Deadlocks



Process A is waiting on Process B

Process B is waiting on Process A

Trajanje Brave

- *Trajanje brave* se odnosi na dužinu vremena za koje DBMS drži bravu.
- Dva parametra utiču na trajanje brave:
  - nivo izolacije
  - dobijanje/otpuštanje intent brave

Nivo Izolacije

- Određuje ponašanje brave za transakciju ili SQL iskaz.
- Standardni SQL definiše 4 nivoa izolacije koji mogu da se postave koristeći SET TRANSACTION ISOLATION LEVEL iskaz:
  - UNCOMMITTED READ
    - aka dirty read
  - COMMITED READ
    - aka cursor stability
  - REPEATABLE READ
  - SERIALIZABLE

Specifikacija dobijanja/otpuštanja intent brave

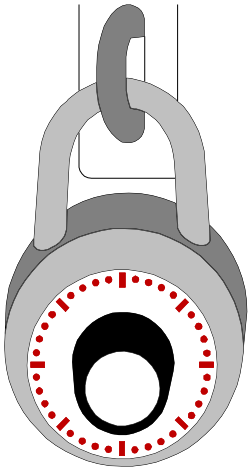
- Utiče na trajanje brave
- Kontrolisati kada se dobijaju i otpuštaju
  - Intent brave mogu da se dobijaju odmah kada transakcija zahteva ili u trenutku kada se transakcija izvršava
  - Intent brave mogu da se otpuste kada se transakcija kompletira ili kada nije ni jedna intent brava nije zahtevana za jedinicu posla.

Eskalacija Brave

- *Eskalacija brave* je proces povećavanja granularnosti brave za proces ili program
- Tipično kontrolisan sistemskim parametrima i DDL parametrima u CREATE iskazu.
- ALTER TABLE some_table SET (LOCK_ESCALATION = TABLE)
- Na primer:
  - Ako je dostignut prag brava koje drži proces (ili ceo DBMS), brave stranice (ili reda) mogu da eskaliraju u brave tabele
  - Mogu da izazovu probleme konkurencije
    - Ukoliko je uzeta brava nad celom tabelom, drugi procesi ne mogu da pristupaju podacima

Tehnike programiranja za minimiziranje problema brava

- Izbegavati deadlokove pisanjem update naredbe u istoj sekvenci, bez obzira na program
  - Na primer, alfabetski redosled po imenu tabele
- Izdati SQL iskaz što bliže kraju rada jedinice posla
  - Što kasnije se desi update, to je kraće trajanje brave



Batch Obrada

- Batch Obrada
  - Kada su programi planirani za izvršavanje u predefinisanom vremenu bez bilo kakvog korisničkog ulaza
- Batch programeri ponekad tretiraju tabele kao fajlove… što NIJE dobra ideja.
  - Misliti relaciono umesto obrada fajla
- Planirati i implementirati COMMIT strategiju u svim batch aplikacijama
  - Umesto držanja brava do kraja programa
  - Suprotno ćete se susresti sa mnogo timeout-a

Pitanja