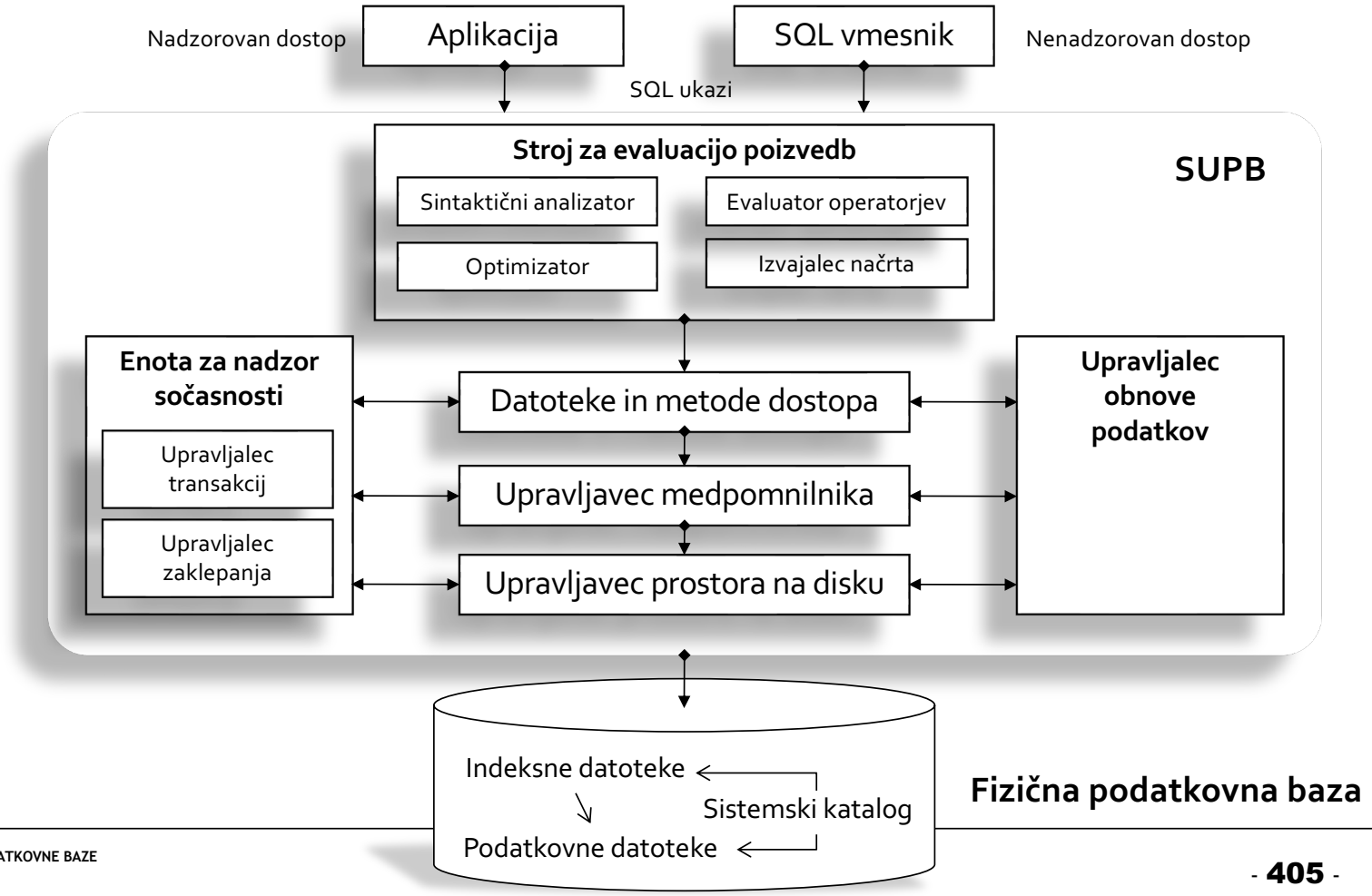
Poglavje VII

Zgradba SUPB in načini dostopa do podatkov

- Zgradba SUPB
- Skupine uporabnikov SUPB
- Različni načini dostopa do podatkov v PB
- Programski dostop do podatkov v PB
- ODBC
- JDBC

Tipična zgradba podatkovnega sistema



Fizična podatkovna baza

Tipična zgradba SUPB...

- Funkcije posameznih enot SUPB:
  - Stroj za evaluacijo poizvedb (Query Evaluation Engine)
    - Sintaksni analizator (Parser): Sintaktično analizira poizvedbo, ki jo SUPB-ju posreduje aplikacija.
    - Optimizator (Optimizer): Na podlagi informacij o tem, kako so podatki shranjeni, izdela učinkovit načrt za izvajanje poizvedbe. Načrt izvajanja predstavlja načrt za izvedbo poizvedbe in je ponavadi predstavljen kot drevo relacijskih operatorjev.
    - Evaluator operatorjev (Operator Evaluator): Na osnovi načrta izvajanja analizira poizvedbo in načrt po potrebi dopolni.
    - Izvajalec načrta (Plan Executor): Izvede poizvedbo po navodilih načrta poizvedbe.

Tipična zgradba SUPB...

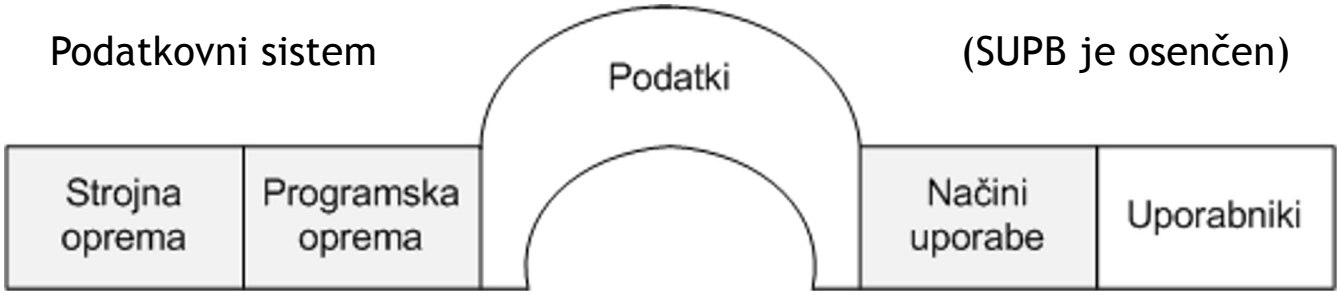
- Funkcije posameznih enot SUPB (nadaljevanje):
  - Datoteke in metode dostopa (Files and Access Methods): enota, ki omogoča delo z datotekami.
  - Upravljalec medpomnilnika (Buffer Manager): Prenaša strani iz diska v pomnilnik glede na bralne potrebe.
  - Upravljalec prostora na disku (Disk Space Manager): Najnižji nivo SUPB je zadolžen za upravljanje z diskom. Vse operacije višjih plasti se tukaj prevedejo v nizko-nivojske ukaze za delo z diskom.

Tipična zgradba SUPB...

- Funkcije posameznih enot SUPB (nadaljevanje):
  - Enota za nadzor sočasnosti (Concurrency Control):
    - Upravljalec transakcij (Transaction Manager): Zagotavlja zaseganje podatkov z uporabo določenih protokolov in skrbi za razporejanje izvajanja transakcij.
    - Upravljalec zaklepanja (Lock Manager): Vzdržuje informacije o zahtevanih in odobrenih zaseženjih podatkov.
  - Upravljalec obnove podatkov (Recovery Manager): Vzdržuje dnevnik in skrbi za obnavljanje sistema v zadnje skladno stanje pred nesrečo.

SUPB in načini dostopa do podatkov

- SUPB: kompleksna zbirka programov, ki v okviru podatkovnega sistema skrbijo za podatke in zagotavlja uporabnikom dostop do njih.
- Glavni nalogi SUPB:
  - upravljanje s podatkovno bazo glede na potrebe **različnih skupin** uporabnikov
  - skrb za razpoložljivost in celovitost shranjenih podatkov.



Uporabniki podatkovne baze

- Uporabniki uporabljajo SUPB na najrazličnejše načine. Glede na vloge, v katerih nastopajo, jih delimo na nekaj tipičnih skupin:
  - Naivni uporabniki
  - Parametrični uporabniki
  - Menujsko vodeni uporabniki
  - Povpraševalni uporabniki
  - Uporabniški programerji
  - Sistemski programerji
  - Administrator(ji) podatkovne baze

Uporabniki podatkovne baze

- Naivni uporabniki
  - občasen dostop do podatkovne baze
  - namenske, enostavne aplikacije (tudi glede interakcije s podatkovno bazo), ki pretežno temeljijo na obrazcih.
  - npr. spletne aplikacije (eBay, …)

Uporabniki podatkovne baze

- Parametrični uporabniki
  - dostopajo do podatkovne baze z uporabo aplikacij, napisanih v splošnonamenskih programskih jezikih
  - pri zagonu teh programov je potrebno po navadi specificirati vrsto parametrov oziroma vhodnih podatkov
  - delo s programi je preprosto, interakcija s podatkovno bazo pa lahko **poljubno zapletena**,
  - zakrivajo kompleksnost dejanskih operacij
  - ščitijo podatkovno bazo pred morebitnimi napačnimi vnosi podatkov (kontrola vhodnih podatkov) in postopki dela
  - uporaba predvsem pri rutinskih uporabah podatkovne baze (npr. v bančništvu, rezervacijskih sistemih, …)

Uporabniki podatkovne baze

- Menujsko vodeni uporabniki
  - dostopajo do podatkov s pomočjo menujsko vodenega dialoga pod nadzorom SUPB, ki korak za korakom gradi poizvedbo
  - le občasno potrebujejo dostop do podatkov in zato niso podrobneje seznanjeni s funkcijami in lastnostmi SUPB
  - predvsem potrebujejo dostop do podatkov, le redko tudi ažuriranje
  - podatkovne potrebe so nepredvidljive in spontane, tako da jih ni možno reševati z vnaprej pripravljenimi uporabniškimi programi

Uporabniki podatkovne baze

- Povpraševalni uporabniki
  - SUPB uporabljajo pogosto in na različne načine
  - za dostop do podatkov pa uporabljajo povpraševalne jezike SUPB (predvsem SQL)
  - poznajo tako ukaze jezika, kot tudi strukturo in vsebino podatkovne baze.
  - ukaze povpraševalnega jezika uporabnik zaporedoma interaktivno posreduje SUPB, ali pa jih (pri kompleksnejših povpraševanjih) zbere v ukazni datoteki, ki jo posreduje SUPB v paketno (ang. batch) izvajanje.

Uporabniki podatkovne baze

- Uporabniški programerji
  - pišejo programe za naivne in parametrične uporabnike glede na njihove potrebe in zahteve
  - glede na pogosto ponavljajoče se dostope v podatkovno bazo, je pomembno zagotoviti učinkovitost teh programov
  - običajno pisani v splošnonamenskih programskih jezikih, ki omogočajo bistveno hitrejše izvajajnje od programov pisanih v povpraševalnih jezikih.
  - dostop do podatkovne baze preko ustreznih programskih vmesnikov

Uporabniki podatkovne baze

- Sistemski programerji
  - vzdržujejo SUPB po navodilih proizvajalca
  - razvijajo splošnonamenske programe in aplikacije za vse uporabnike podatkovne baze.
  - razvijajo in vzdržujejo tudi spletne in aplikacijske vmesnike za menujsko vodene uporabnike (z orodji proizvajalca)

Uporabniki podatkovne baze

- Skrbnik podatkovne baze (DBA) skrbi za razpoložljivost, celovitost in uporabnost podatkov v podatkovni bazi. Poglavitne naloge DBA so:
  - definiranje in ažuriranje notranjih, konceptualnih in zunanjih shem
  - kreiranje in inicializacija fizične podatkovne baze
  - razvoj in vzdrževanje programskih orodij za podporo končnim uporabnikom in uporabniškim programerjem
  - zaščita podatkovne baze pred nesrečami in njeno obnavljanje
  - postopki za vzdrževanje kvalitete podatkovne baze
  - upravljanje sistema gesel in dostopnih dovoljenj za uporabnike

Uporabniki podatkovne baze

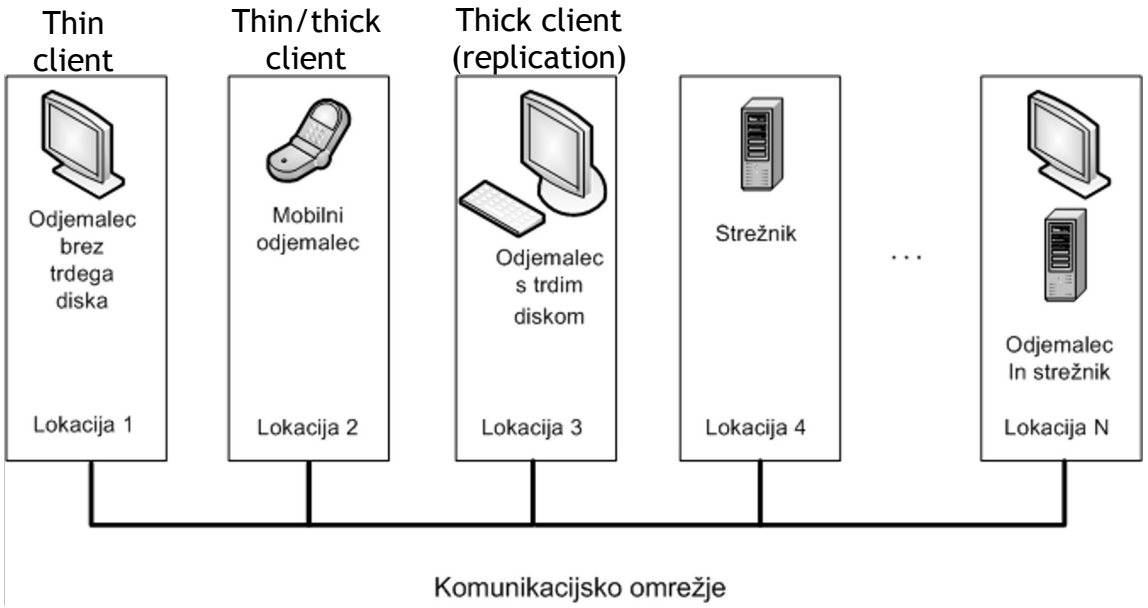
- Skrbnik podatkovne baze (DBA)
  - nadzorovanje zmogljivosti in uporabe podatkovne baze ter izvajanje ustreznih reorganizacij in prilagoditev (uglaševanje)
  - pomoč uporabnikom pri načrtovanju in uporabi podatkov ter uporabi programskih orodij v okviru SUPB
  - administratorske naloge lahko opravlja ena ali več oseb
  - v manjših okoljih se lahko nekatere naloge upravitelja podatkovne baze, kot so kreiranje shem in izdaja pristopnih dovoljenj za lastne podatke prenesejo tudi na končne uporabnike

Ponovitev: uporabniki podatkovne baze

- Uporabniki uporabljajo SUPB na najrazličnejše načine. Glede na vloge, v katerih nastopajo, jih delimo na nekaj tipičnih skupin:
  - Naivni uporabniki
  - Parametrični uporabniki
  - Menujsko vodeni uporabniki
  - Povpraševalni uporabniki
  - Uporabniški programerji
  - Sistemski programerji
  - Administrator(ji) podatkovne baze

Dostop do podatkovne baze

- Različne skupine uporabnikov do podatkov dostopajo na različne načinov (preko podatkovnih vmesnikov ali podatkovnih jezikov)



Dostop do podatkov: podatkovni vmesniki

- Menujski vmesniki:
  - aplikacije za menujske uporabnike,
  - vodenje korak za korakom
  - dejanski ukaz se gradi postopoma
- Obrazci (forms):
  - Naivni uporabniki
  - Vnos in spreminjanje podatkov
  - Kontrola podatkov in z njimi povezane transakcije
- Grafični uporabniški vmesniki
  - Podatkovna baza kot simboličen diagram podatkovnih objektov
  - Uporabnik določi operacijo s klikanjem po diagramu (QBE)

Dostop do podatkov: podatkovni vmesniki

- Vmesniki z uporabo naravnega jezika
  - Postavljanje vprašanj v poenostavljenem naravnem jeziku (običajno angleščini)
  - Potrebna pazljivost zaradi možnih dvoumnosti
- Vmesniki za parametrične uporabnike
  - Relativno majhno število pogostih operacij
  - Možnost nastavljanja parametrov delovanja
  - Hiter zagon strogo namenskih aplikacij (bližnjice, kombinacije tipk)
- Vmesniki za skrbnike PB
  - Izvajanje privilegiranih ukazov
  - Vpogled v delovanje SUPB, možnost reorganizacije

Dostop do podatkov: podatkovni jeziki

- Povpraševalni uporabniki in programerji
- Nizkonivojski postopkovni
  - Bolj ali manj splošnonamenski programski jeziki
  - Zapisno (vrstično) usmerjeni: podatke definirajo in do njih dostopajo preko programskih konstruktov (zank)
  - V uporabi le še v starih (legacy) aplikacijah, npr. COBOL
- Visokonivojski nepostopkovni (npr. SQL)
  - DDL in DML: samostojna uporaba za opis kompleksnih operacij znotraj SUPB
  - Možnost vključevanja (embedding) v splošnonamenske programske jezike s posebnimi orodji
  - Uporaba v splošnonamenskih programskih jezikih z uporabo programskih vmesnikov

Programski dostop do podatkovne baze

- Dostop do podatkov iz poljubnega programa oz. programskega jezika (odjemalca):
  - Samostojne aplikacije – tudi mobilne (Python, C, Java, …)
  - Spletne aplikacije (PHP, Ruby, .NET, Java, Python, …)
  - Nekateri splošnonamenski programi (Excel, Access, Word, …)
- Vsak SUPB definira lastne protokole za čim-učinkovitejšo komunikacijo odjemalec/strežnik
- Vključeni (embedded) SQL: predprocesirana programska koda
- Obstoj splošnih komunikacijskih standardov

Vključeni (embedded) SQL

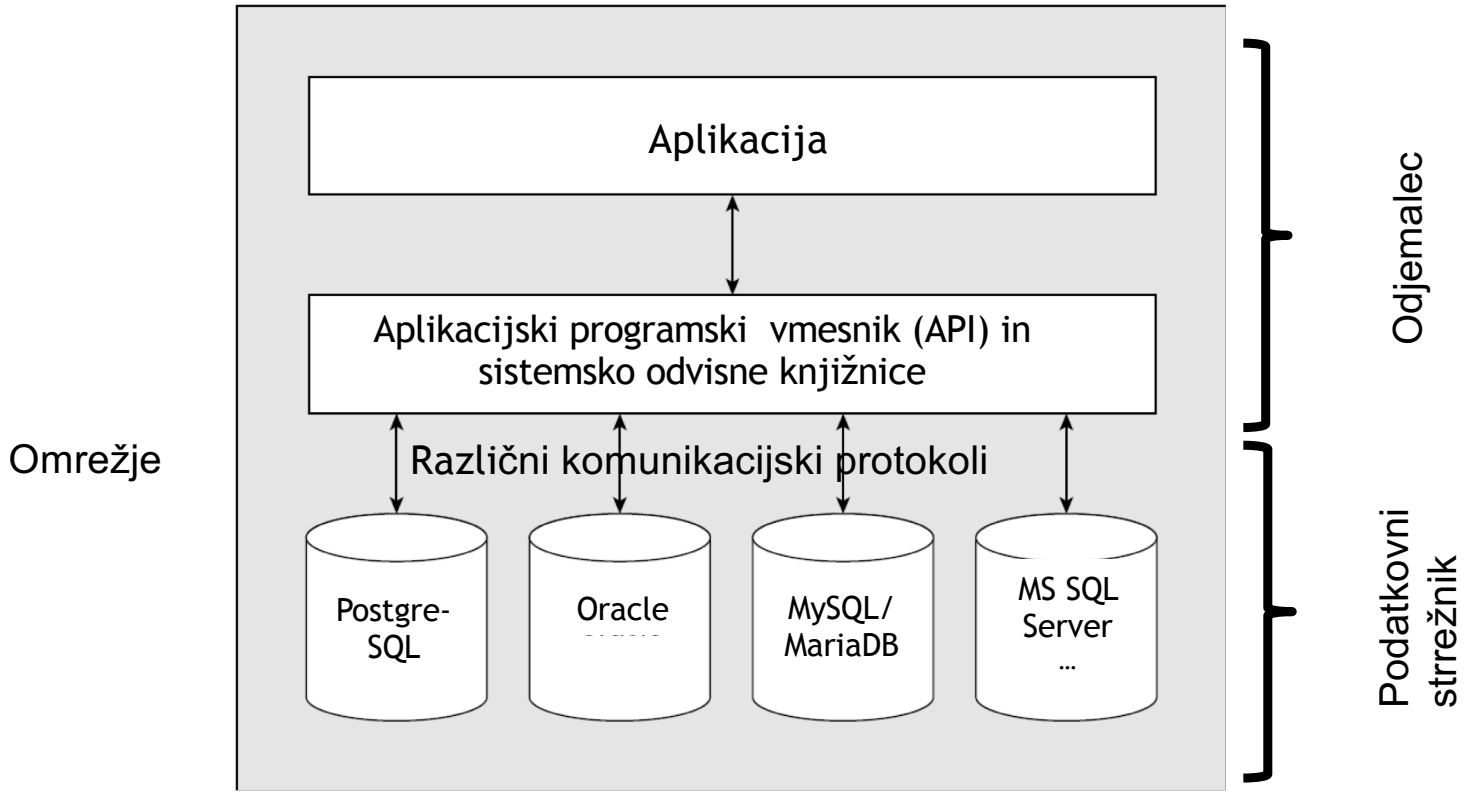
- Posebni predprocesorji podprti s strani proizvajalca SUPB preoblikujejo izvorno kodo in jo posredujejo prevajalniku
- ISO standard
- Primer v programskem jeziku C

```
EXEC SQL BEGIN DECLARE SECTION;
       int       jid;
       int       cid;
       VARCHAR    dan[20];
EXEC SQL END DECLARE SECTION;

int main()
{
   jid = 49;
   cid = 103;
   dan="2010-04-04";

   EXEC SQL INSERT INTO
        rezervacija(jid, cid, dan)
           VALUES (:jid, :cid, TO_DATE (:dan));
   EXEC SQL COMMIT WORK;
}
```

Komunikacija od aplikacije do podatkovne baze



Nastanek standardnih programskih vmesnikov

- Različni proizvajalci podatkovnih baz uporabljajo različne protokole in programske vmesnike (API)
- Težavno programiranje aplikacij
- Leta 1992 se pojavi vmesnik ODBC (open data base connectivity), ki skuša poenotiti programski dostop
- Aplikacije prenosljive na različne platforme, vendar je njihova funkcionalnost in učinkovitost nekoliko okrnjena v primerjavi z uporabo originalnih programskih vmesnikov

ODBC - open data base connectivity

- Nastal je leta 1992 v sodelovanju podjetij Microsoft in Simba Technologies
- Leta 1995 je ODBC 3.0 postal del standarda ISO/IEC 9075-3 -- Information technology -- Database languages -- SQL -- Part 3: Call-Level Interface (SQL/CLI).
- Sodobni ODBC standard (4.x v razvoju od 2016 dalje) sloni na različnih standardnih Call Level Interface (CLI) specifikacijah iz SQL Access Group, X/Open in ISO/IEC

ODBC

- Prevzeli so ga vsi pomembnejši proizvajalci SUPB
- Množica implementacij ODBC gonilniških sistemov za različne operacijske sisteme in SUPB-je:
  - Microsoft ODBC (vgrajen v Windows, tudi DAO, DAC: data access objects, data access components),
  - iODBC (open source: MacOS, Linux, Solaris, ...),
  - UnixODBC (open source: Linux),
  - IBM i5/OS (IBM, DB2)
- Gonilniki proizvajalcev SUPB (OS $\times$ SUPB)
  - Oracle, MS, IBM, PostgreSQL, MariaDB in MySQL, …

Značilnosti ODBC

- Proceduralni programski vmesnik za dostop do podatkovne baze (v osnovi v obliki C knjižnic)
- Omejitev ODBC: delo z SQL standardom, kot ga definira ODBC
- Težaven dostop do specifičnih razširitev SQL: omogočen s pomočjo meta-podatkovnih funkcij
- Kaj potrebujemo za delo:
  - ODBC aplikacijski vmesnik za odjemalčev OS
  - ODBC gonilnik za strežniški OS in uporabljan SUPB

Zakaj ODBC?

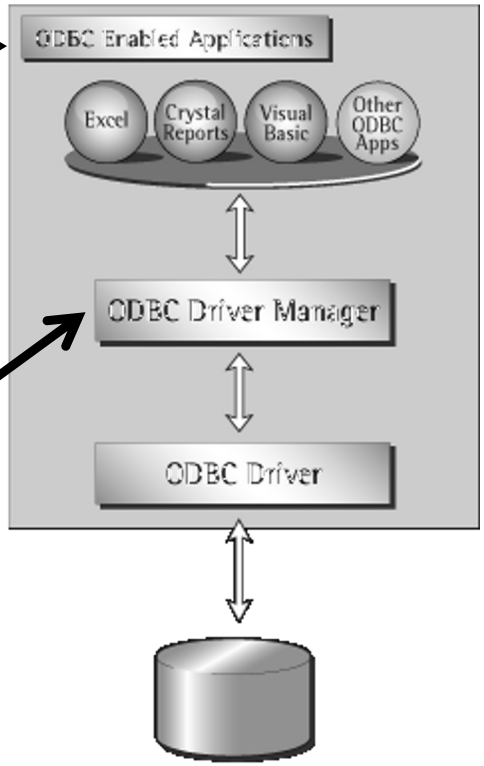
- Aplikacije niso vezane na konkreten API
- SQL stavke lahko v kodo vključujemo statično ali dinamično
- Aplikacij ne zanima dejanski komunikacijski protokol
- Format podatkov prilagojen programskemu jeziku
- Standardiziran vmesnik (X/Open, ISO CLI)
- Univerzalno sprejet in podprt
- Specifikacija in nadaljnji razvoj: https://github.com/Microsoft/ODBC-Specification

Kaj nam ODBC ponuja

- Knjižnico funkcij, ki omogoča aplikaciji povezavo s SUPB, izvajanje SQL stavkov in dostop do rezultatov in statusa izvajanja
- Standarden način za prijavo in odjavo na SUPB
- Standardno (a omejeno) predstavitev podatkovnih tipov
- Standarden nabor sporočil o napakah
- Podporo SQL sintaksi po X/Open in ISO CLI specifikacijah

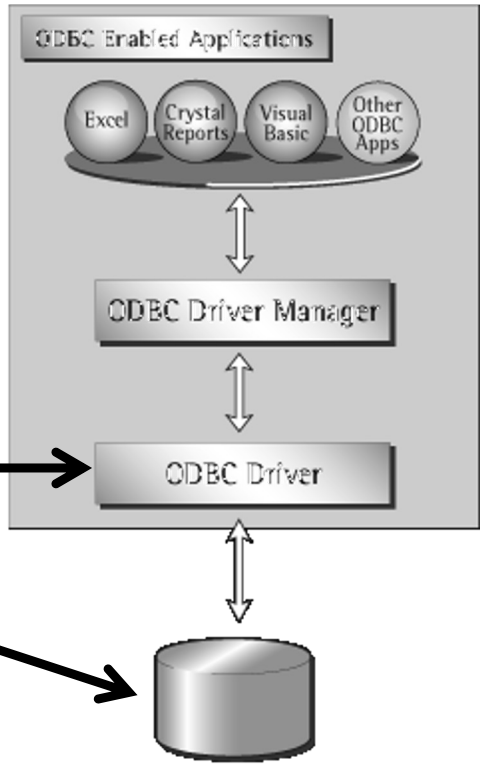
Arhitektura ODBC

- Aplikacije:
  - procesiranje podatkov,
  - klici ODBC funkcij za posredovanje poizvedb in rezultatov
- ODBC upravljalec gonilnikov:
  - Nalaga gonilnike glede na potrebe aplikacij
  - Procesira klice ODBC funkcij in jih posreduje gonilniku



Arhitektura ODBC

- ODBC gonilnik:
  - Prevzema klice ODBC funkcij, jih po potrebi preoblikuje in posreduje SUPB
  - Omogoča manjkajočo funkcionalnost glede na implementiran ODBC standard
- Podatkovni vir:
  - SUPB
  - tekstovne datoteke
  - preglednice
  - …



ODBC in standardni SQL

- ODBC standardizira tako aplikacijski vmesnik (API) kot tudi podporo SQL ukazom
- Popolna podpora od ODBC 3.0 dalje:
  - Minimalni SQL
  - Standardni SQL (X/Open, ISO CLI)
  - Razširjeni SQL
- Razvoj ODBC 4.0 najavljen leta 2016
  - Delo z delno strukturiranimi in hierarhičnimi podatki
  - Prilagajanje spletnim aplikacijam (avtentikacija, …)

ODBC in standardni SQL

- Minimalni SQL
  - Data Definition Language (DDL): CREATE TABLE in DROP TABLE
  - Data Manipulation Language (DML): enostavni SELECT, INSERT, UPDATE, in DELETE z iskalnim pogojem
  - Preprosti izrazi: (npr. as A>B+C)
  - Samo znakovni podatkovni tipi: CHAR, VARCHAR, LONG VARCHAR (prenos podatkov)

ODBC in standardni SQL

- Standardni SQL
  - Vsebuje minimalni SQL
  - Data Definition Language (DDL): ALTER TABLE, CREATE INDEX, DROP INDEX, CREATE VIEW, DROP VIEW, GRANT, in REVOKE
  - Data Manipulation Language (DML): polni SELECT stavek
  - Izrazi: gnezdene poizvedbe, skupinski operatorji (npr. SUM, MIN, …)
  - Podatkovni tip: DECIMAL, NUMERIC, SMALLINT, INTEGER, REAL, FLOAT, DOUBLE PRECISION

ODBC in standardni SQL

- Razširjeni SQL
  - Minimalni in osnovni SQL
  - Data Manipulation Language (DML): zunanji stiki, pozicijski UPDATE, pozicijski DELETE, SELECT FOR UPDATE, unije
  - Izrazi: skalarne funkcije (npr.SUBSTRING, ABS), določila za deklaracijo konstant DATE, TIME in TIMESTAMP
  - Podatkovni tipi: BIT, TINYINT, BIGINT, BINARY, VARBINARY, LONG VARBINARY, DATE, TIME, TIMESTAMP
  - Paketi SQL stavkov
  - Podpora shranjenim proceduram (klicanje)

pyodbc – implementacija ODBC za Python

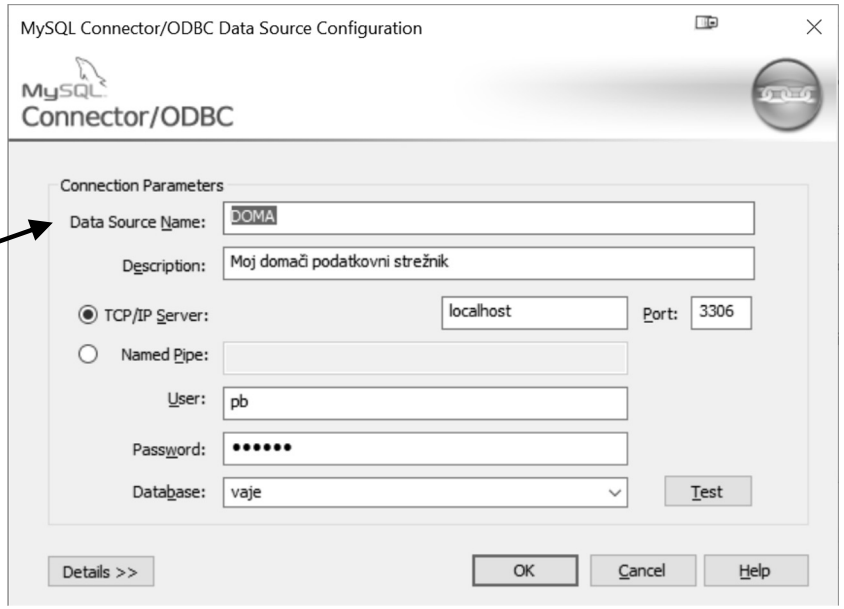
- pyodbc je modul za Python ki omogoča dostop do poljubnega SUPB (ki podpora ODBC)
- implementira Python Database API Specification v2.0 z dodatki, ki poenostavljajo delo s SUPB
  - DB API v2 uporabljaju tudi nekateri drugi komunikacijski moduli (npr. pyMySQL, mysqlclient, psycopg2, cx_oracle, ki pa implementirajo "native" protokol)
- pyodbc je odprtokoden, uporablja MIT licenco, in ga lahko zastonj uporabljamo tako v pridobitne, kot nepridobitne namene (vključno z izvorno kodo)
- GitHub stran in dokumentacija (originalno Google code): https://mkleehammer.github.io/pyodbc/

Predpriprava na uporabo pyodbc

- SUPB s podatki
- Python (3.x) in pyodbc (za Python 3.x)
- ODBC gonilnik za izbrani OS (**32/64 bit**) in SUPB
- MySQL: Connector/ODBC
  - https://dev.mysql.com/downloads/connector/odbc/
- PostgreSQL: psqlODBC
  - https://odbc.postgresql.org/
- Primer uporabe ODBC iz jezika C:
  - https://www.drdobbs.com/c-database-programming-with-odbc/184403098/

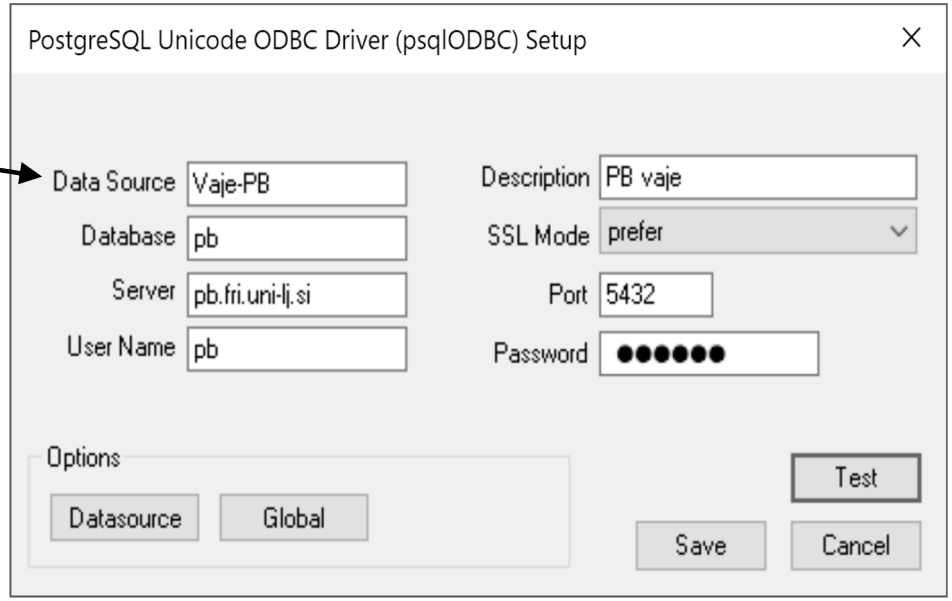
Priprava DSN podatkovnega vira (MySQL)

- Odprite Control Panel->Administrative tools->ODBC Data Sources (64-bit ali 32-bit, odvisno od aplikacije)
- V zavihku User DSN izberite Add in nato določite ODBC gonilnik:
  - MySQL ODBC 8.x Unicode driver
  - Vnesite vrednosti s slike: DSN je lahko poljuben.
  - Lahko vnesete uporab. ime in geslo



Priprava DSN podatkovnega vira (PostgreSQL)

- Odprite Control Panel->Administrative tools-> ODBC Data Sources
- V zavihku User DSN izberite Add in nato določite ODBC gonilnik:
  - PostgreSQL Unicode (priporočeno) ali ANSI
  - Poimenujte DSN
  - Vnesite podatke s slike (pb/pbvaje)
  - Testirajte (gumb Test) in shranite vir



Osnovni gradniki pyodbc

- Uvoz modula:
  import pyodbc
- Najpomembnejši razredi:
  - Povezava (connection)
  - Kurzor (cursor)
  - Podatkovni tipi in njihovi konstruktorji
  - Obravnava napak

pyodbc: povezava

- Povezavo c ustvarimo z ukazom:
  c = pyodbc.connect(ConnectionString)
- Povezovalni niz (ConnectionString) določa povezavo, npr.

```
ConnectionString = 'DSN=Vaje-PB;UID=pb;PWD=pbvaje'
ali
ConnectionString = 'DSN=DOMA;UID=pb;PWD=pbvaje‘
```

- UID in PWD sta lahko privzeta iz DSN

pyodbc: povezava

- ConnectionString bi lahko napisali tudi brez definiranega DSN:

```
ConnectionString = 'DRIVER={MySQL ODBC 8.0 driver};
                   SERVER=localhost;DATABASE=vaje;UID=pb;PWD=pbvaje;
                   CHARSET=UTF8'
```

- Kako strukturiramo povezovalni niz:
  - https://www.connectionstrings.com/
  - Za MySQL Connector/ODBC: https://dev.mysql.com/doc/connector-odbc/en/connector-odbc-configuration.html

pyodbc: povezava

- Povezava c ponuja metode:
- close(): zapri povezavo, enako pri destruktorju objekta
  - c.close()
- commit(): uveljavi transakcijo (če SUPB podpira)
  - c.commit()
- rollback(): razveljavi transakcijo (če SUPB podpira)
  - c.rollback()
- cursor(): vrne nov kurzorski objekt, ki uporablja povezavo c
  - cursor = c.cursor()

pyodbc: kurzor (iteracija po rezultatih)

- Kurzor x ustvarimo z ukazom:
  x = c.cursor()
- Nekateri atributi:
  - description: opis stolpcev rezultata (shema)
  - rowcount: število vrstic rezultata
- Nekatere metode:
  - execute(ukaz, [parametri]): izvede ukaz z opcijskimi parametri
  - fetchall(): prenese vse vrstice rezultata
  - fetchone(), fetchmany(size): preneseta eno ali več vrstic

pyodbc: kurzor

- Po kurzorju lahko iteriramo, vendar samo enkrat:

```
x.execute(SQLukaz)
for r in x:
    print(r)
```

- Več iteracij:

```
v = x.fetchall()
```

nato lahko poljubno mnogokrat iteriramo po v:

```
for r in v:
    print(r)
```

pyodbc: kurzor in SQL ukazi

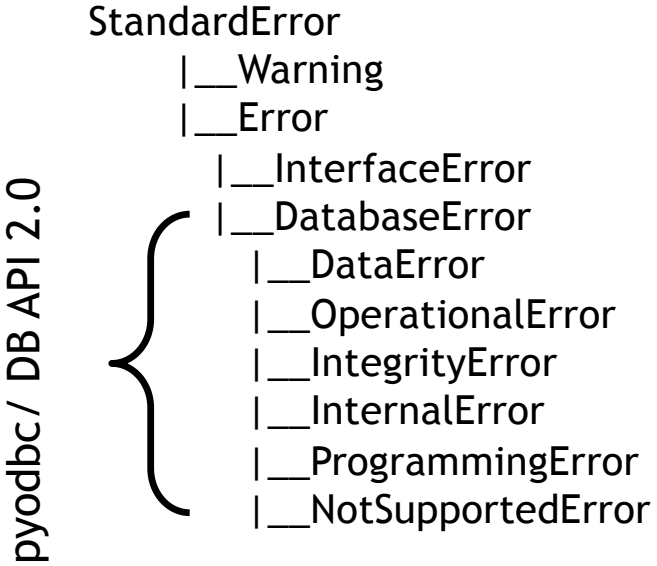
- SQL ukaz je načeloma lahko poljuben niz znakov
- Parametri v SQL ukazih. Primer:

  SELECT * FROM **jadralec**;

  - način parametrizacije z operacijami na nizih – možen „SQL injection"
    - x.execute('SELECT %s FROM %s' % ('*', **'jadralec'**))
    - x.execute('INSERT INTO %s %s' % ('jadralec', "**VALUES(17, 'Peter', 24,9)**")
  - pyodbc/SQL prenos parametrov v metodi execute:
    - ? označuje parameter, seznam parametrov sledi za ukazom
    - x.execute('SELECT ? FROM ?, '*', 'jadralec')
    - x.execute('SELECT ? FROM ?, ('*', 'jadralec'))

pyodbc: obravnava napak

- Razredi pyodbc ob napakah javljajo naslednje izjeme:
  - DatabaseError
  - DataError
  - OperationalError
  - IntegrityError
  - InternalError
  - ProgrammingError
  - NotSupportedError



pyodbc: obravnava napak

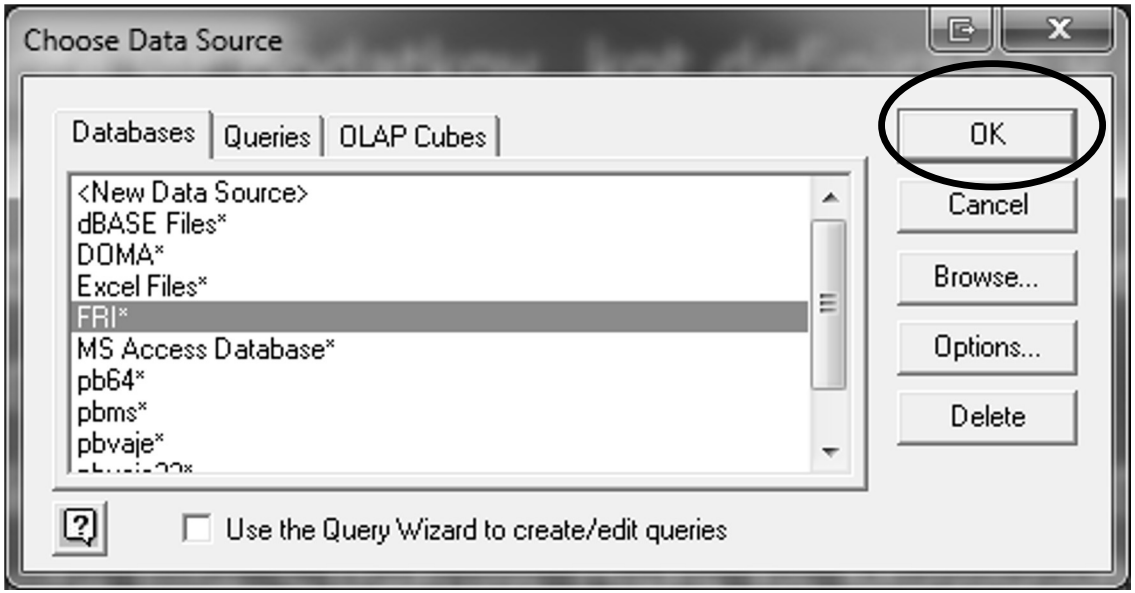
```
try:
        x.execute ( SQLukaz)
        ...
except pyodbc.DataError:
        -- obravnava napake
        pass
...
except pyodbc.DatabaseError:
        -- obravnava napake
        pass
except:
        -- obravnava ostalih napak
        pass
```

pyodbc: preslikava med ODBC/SQL in Pythonovimi podatkovnimi tipi

ODBC	**Python**
char varchar longvarchar GUID	string
wchar wvarchar wlongvarchar	unicode
smallint integer tinyint	int
bigint	long
decimal numeric	decimal
real float double	double
date	datetime.date
time	datetime.time
timestamp	datetime.datetime
bit	bool
binary varbinary longvarbinary	buffer
SQL Server XML type	unicode

Microsoft Excel in ODBC

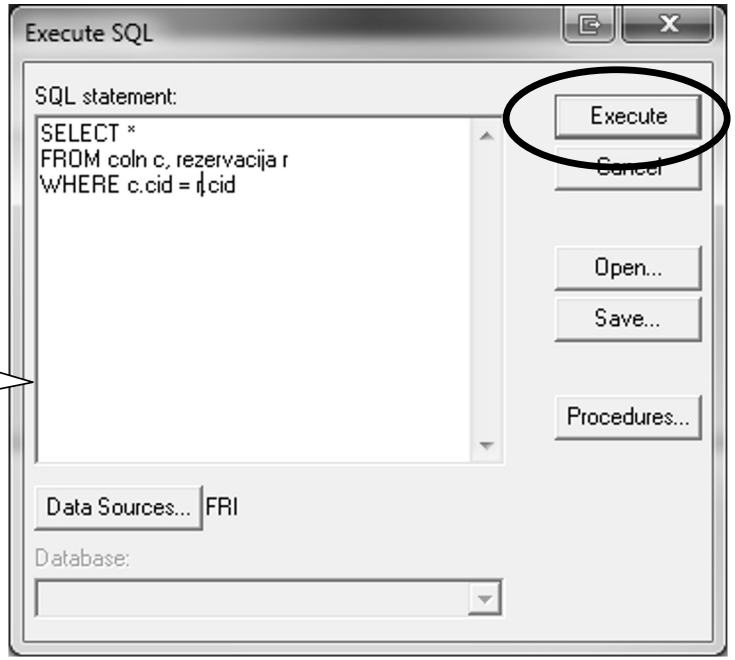
- Izberite Data->Get DataFrom->Other Sources->From Microsoft Query
  - možna je še množica drugih načinov izbire povezave do podatkovnega vira
- Izberite vir podatkov, kot definirano v ODBC Data Sources (npr. FRI) in OK



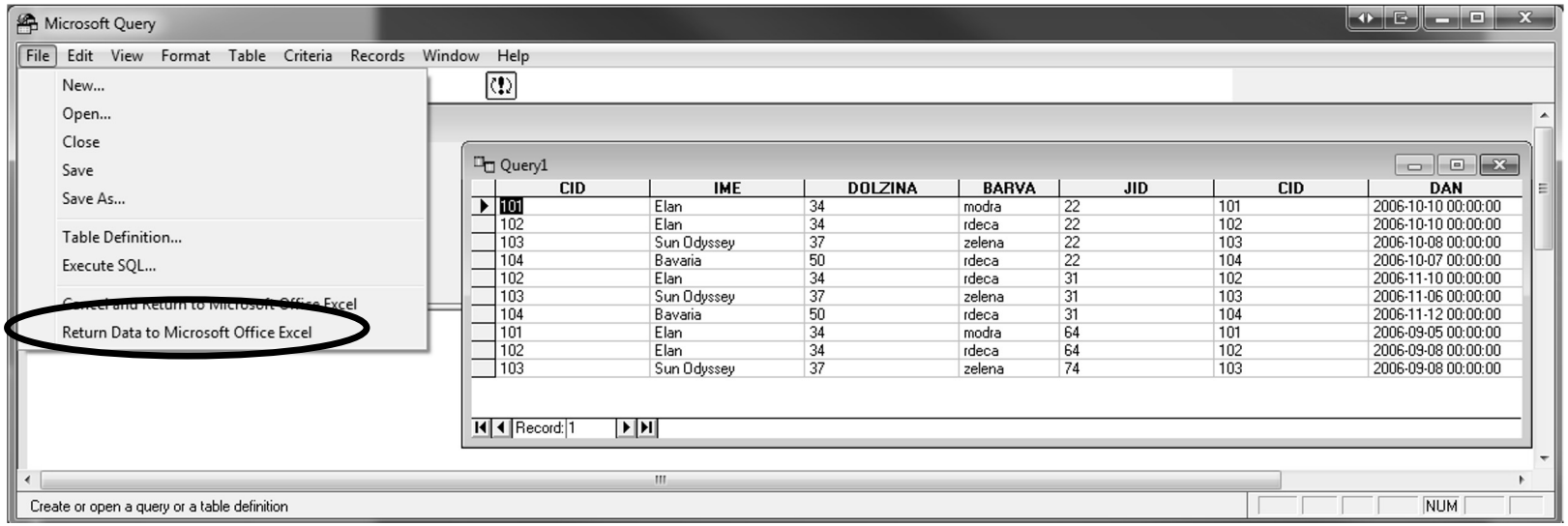
Microsoft Excel in ODBC

- Ne izberite nobene tabele (gumb Cancel), odpre se Microsoft Query
- Ne izberite nobene tabele (gumb Close)
- Izberite File->Execute SQL
- Vnesite SQL poizvedbo in pritisnite gumb Execute

SQL poizvedbo je smiselno napisati in preveriti v za to namenjenem okolju (SQL Developer, MySQL Workbench) ob upoštevanju ODBC omejitev.

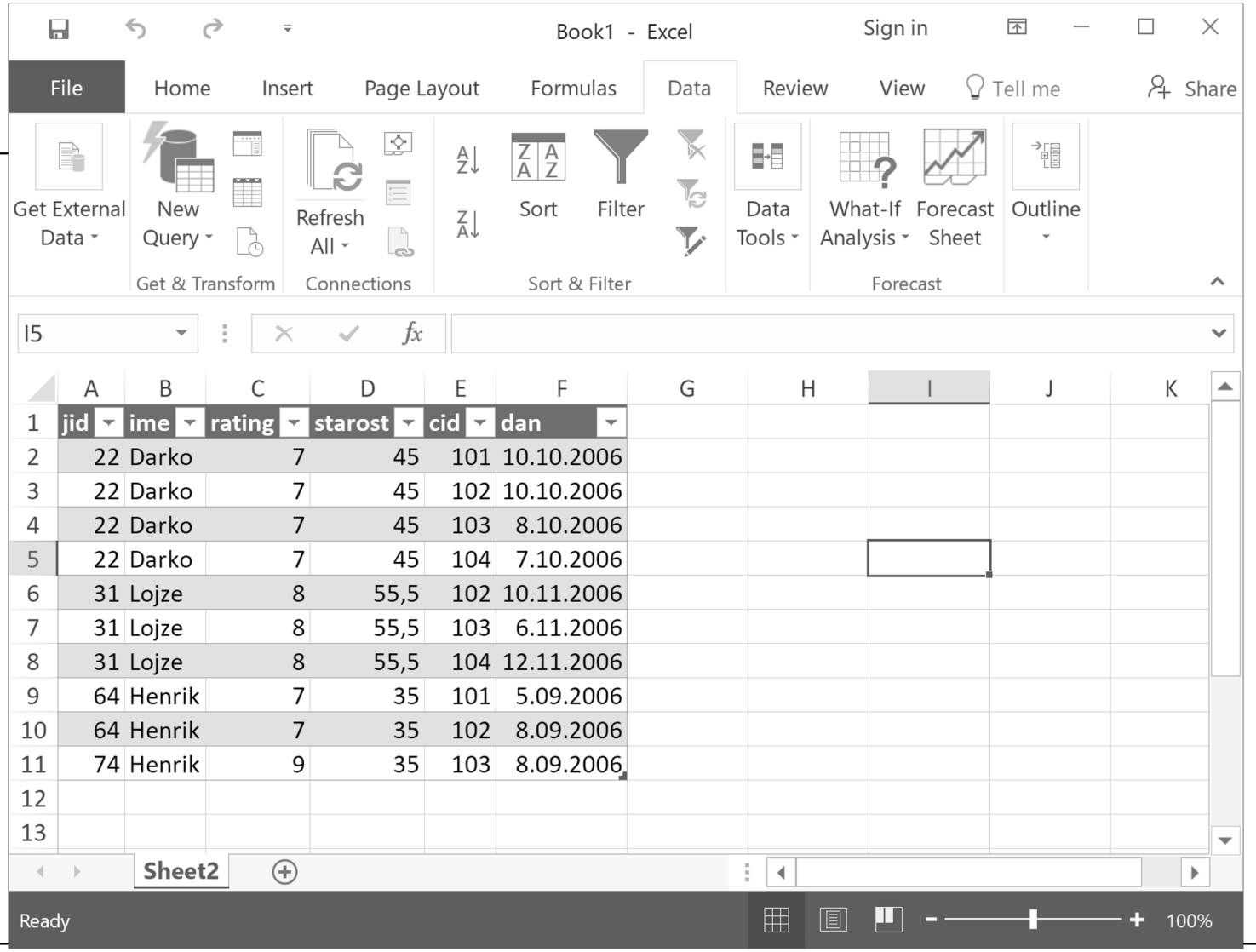


Microsoft Excel in ODBC



- Izberite File->Return Data to Microsoft Excel
- V Excelu dobite tabelo z rezultatom
- Odvisno od definicije DSN (z ali brez gesla) je včasih potrebno vnesti uporbniško ime in geslo za dostop do podatkovne baze

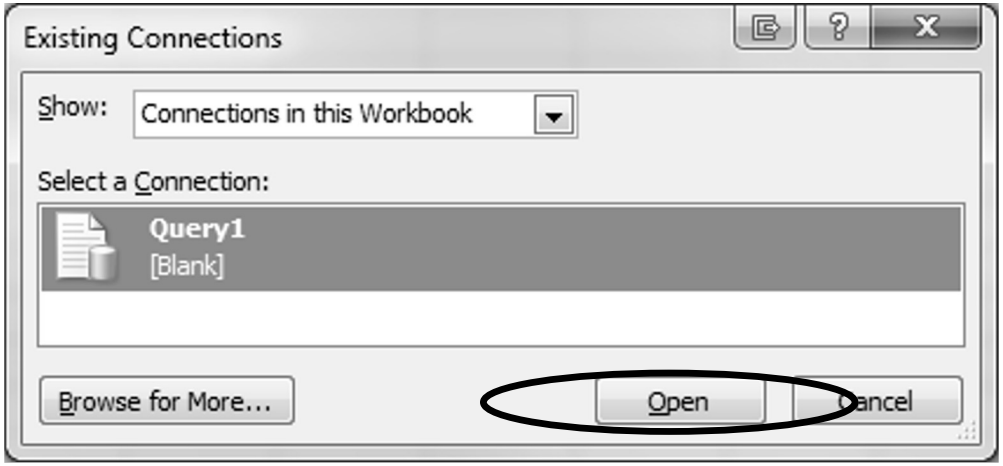
Rezultat poizvedbe v Excelu



	A	B	C	D	E	F
1	jid	ime	rating	starost	cid	dan
2	22	Darko	7	45	101	10.10.2006
3	22	Darko	7	45	102	10.10.2006
4	22	Darko	7	45	103	8.10.2006
5	22	Darko	7	45	104	7.10.2006
6	31	Lojze	8	55,5	102	10.11.2006
7	31	Lojze	8	55,5	103	6.11.2006
8	31	Lojze	8	55,5	104	12.11.2006
9	64	Henrik	7	35	101	5.09.2006
10	64	Henrik	7	35	102	8.09.2006
11	74	Henrik	9	35	103	8.09.2006
12						
13						

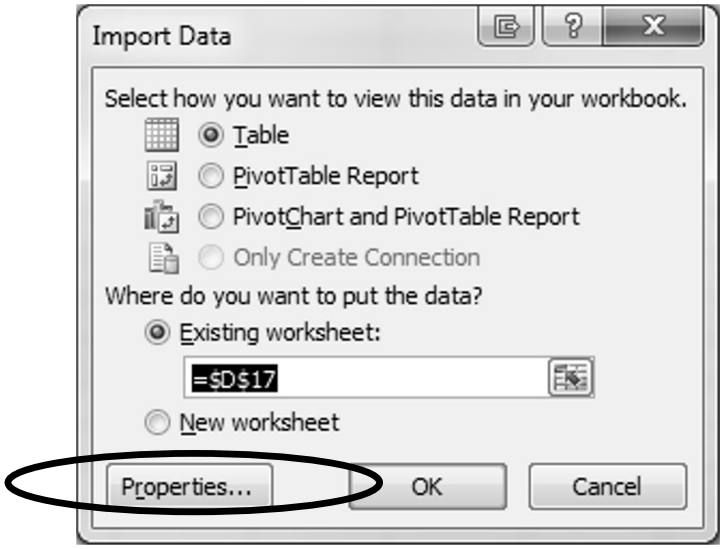
Microsoft Excel in ODBC

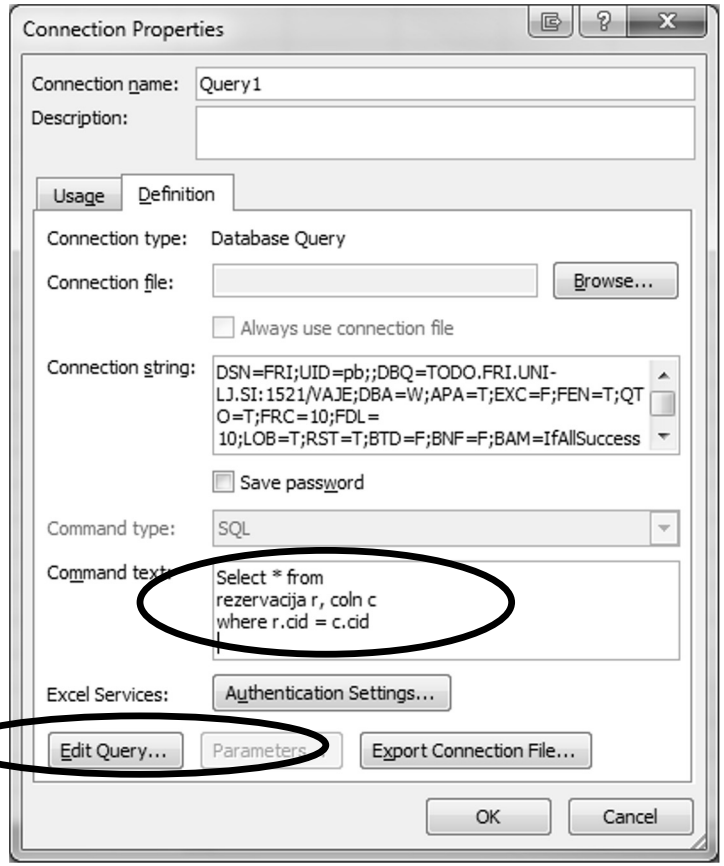
- Povezave s podatkovno bazo so "žive"
  - Osveževanje rezultata poizvedbe ne poteka avtomatsko
  - S pritiskom na Data->Refresh All osvežimo rezultat
- Urejanje poizvedb
  - Pritisnite Data->Existing Connections
  - Izberite ustrezno poizvedbo in pritisnite Open



Microsoft Excel in ODBC

- Izberite Properties in nato zavihek Definition
- V okencu Command text ali s klikom na Edit Query lahko popravimo poizvedbo





Python in SQLite

- SQLite https://www.sqlite.org/index.html:
  - Vgrajen (embedded) SUPB
  - Kompleten SUPB v knjižnici
  - PB hrani v pomnilniku ali datotekah
  - Dobra podpora standardom SQL
  - Široka sprejetost (Android, iOS, …)
  - Binarna kompatibilnost podatkovnih datotek
- Python: modul sqlite3 je del standardne knjižnice (Python 2.7 in 3.x)
- Podpira *Python Database API Specification v2.0* (kot pyODBC)
- Elegantna uporaba za delno replikacijo podatkov (thick client)

Python in SQLite

- Uporaba
  - Kot pyODBC, le drugačna vzpostavitev povezave

```
# Baza bo v pomnilniku
db = sqlite3.connect(':memory:')
# Baza je/bo v datoteki (odpre ali kreira)
db = sqlite3.connect('pb/moja_baza')
```

  - Obvezno potrjevanje sprememb (ekspliciten commit)

```
db.commit()
```

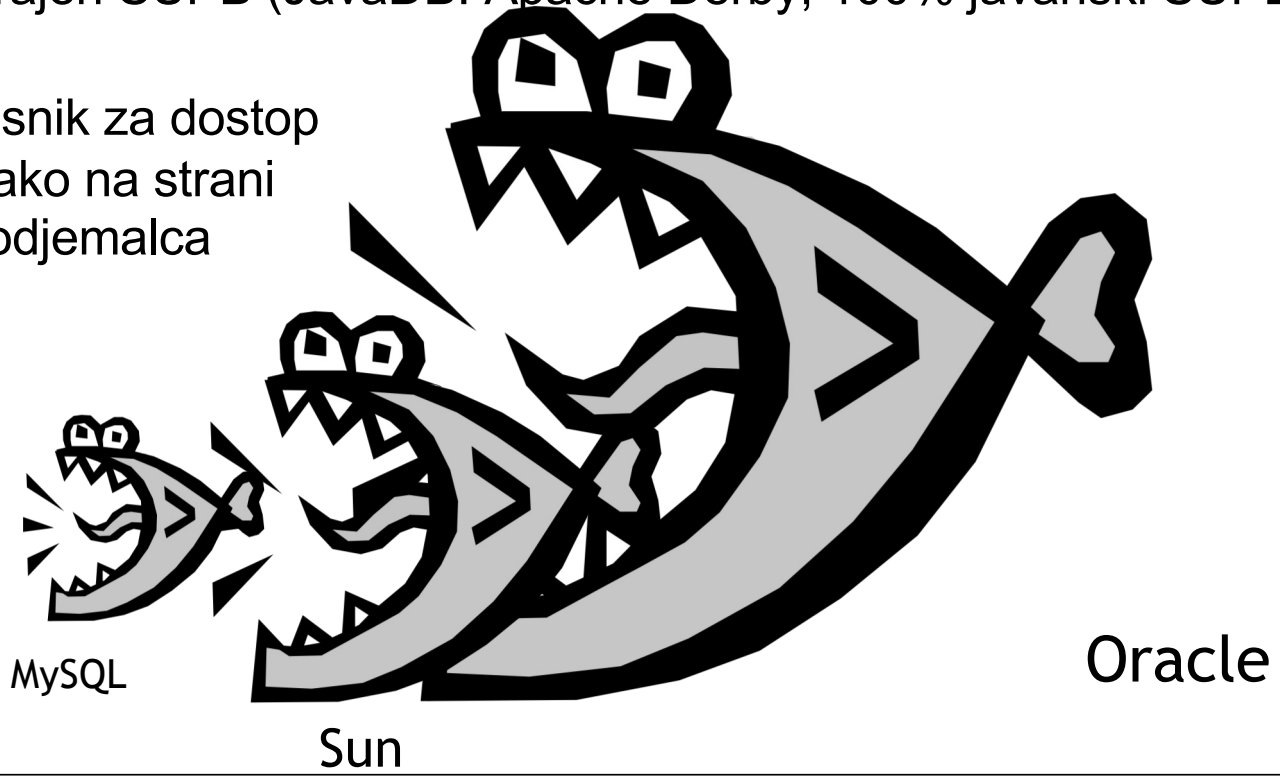
  - Obvezno zapiranje povezave

```
db.close()
```

- Primeri v delovnem zvezku

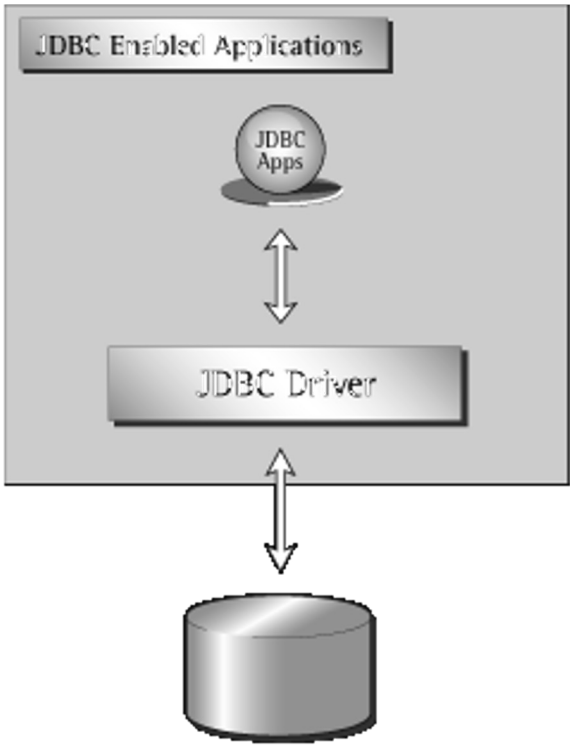
Java in dostop do podatkovnih baz

- Sun Java: že od vsega začetka namenjena pisanju poslovnih aplikacij in apletov
- Enostaven vgrajen SUPB (JavaDB: Apache Derby, 100% javanski SUPB) kmalu ni več dovolj
- Potreben vmesnik za dostop do podatkov tako na strani strežnika kot odjemalca



Java Database Connectivity - JDBC

- Objektno usmerjena implementacija v duhu ODBC
- Arhitektura praktično identična ODBC
- Aplikacija potrebuje JDBC aplikacijski vmesnik (del javanskega standarda) in JDBC gonilnik za konkretno bazo
- V Javi 1.4 se pojavi JDBC 3.0
- V Javi 6 se pojavi JDBC 4.0
- Trenutna verzija (2022) 4.3 "maintenance release 3" je vključena od verzije Java SE 9 dalje



Uporaba JDBC gonilnikov

- Kaj potrebujemo za delo:
  - JDBC razrede (java.sql.*)
  - JDBC gonilnik (.jar arhiv, lokacija odvisna od proizvajalca SUPB)
- Tipi JDBC gonilnikov
  - Tip 1, JDBC-ODBC most: preslika klice JDBC metod v ODBC klice procedur. Primerno le kadar nimamo JDBC gonilnika.
  - Tip 2, Native-API: kliče direktno API gonilnika za PB na našem sistemu. Neprenosljivo.
  - Tip 3, network-protocol: preko mreže dostopa z SUPB-neodvisnim protokolom do posrednika na strežniku (middleware), ki protokol pretvori v takšne klice, ki jih ciljni SUPB razume
  - Tip 4, native-protocol: direktno preko mreže dostopa do SUPB z uporabo njemu lastnega protokola
- Tipa 1 in 2 sta hibridna (Java/ strojna koda), tipa 3 in 4 pa sta pisana v čisti Javi in zato maksimalno prenosljiva.

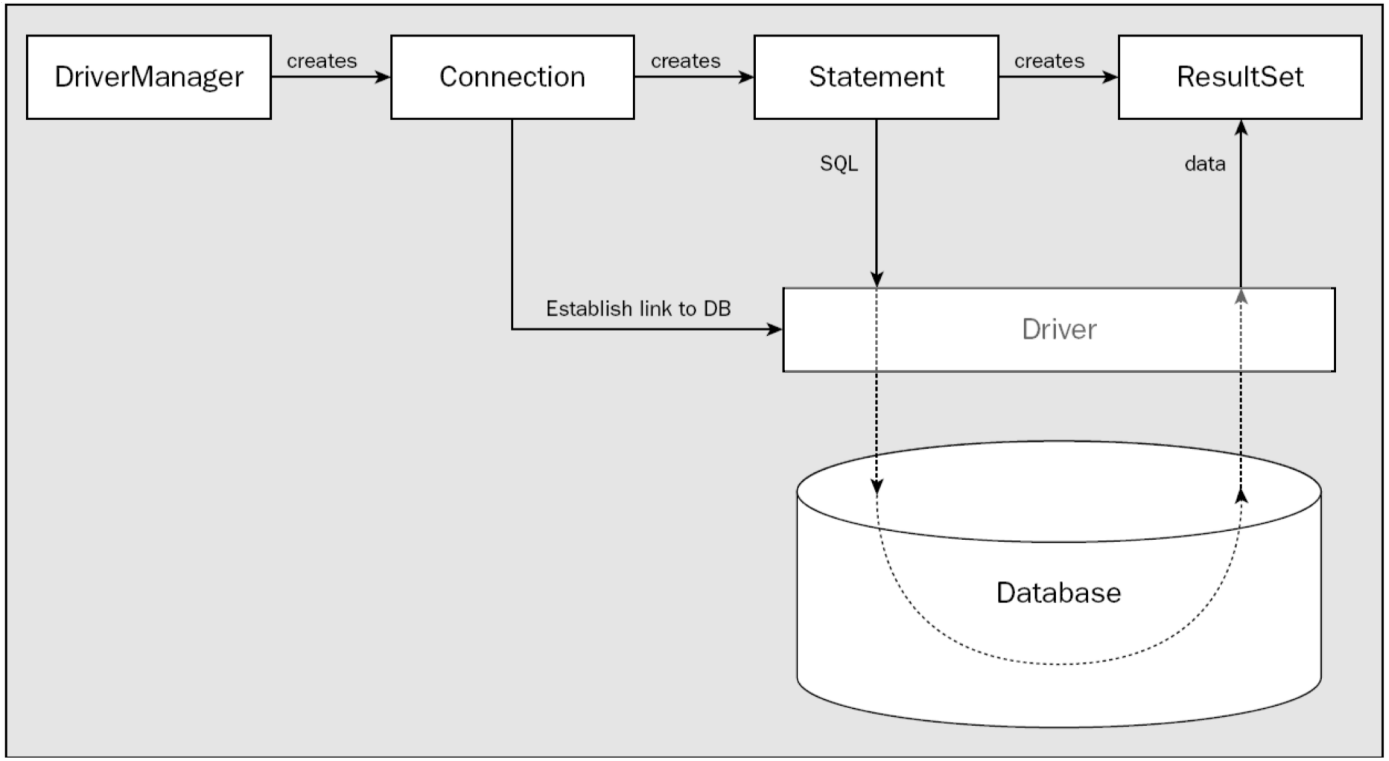
Dostopnost JDBC gonilnikov

- Ponuja jih večina proizvajalcev sodobnih SUPB in tudi nekateri samostojni razvijalci
- Baza JDBC gonilnikov:
  http://www.databasedrivers.com/jdbc/
- Nekateri veliki proizvajalci:
  - Oracle: zraven klienta, JDBC tip 2 ali 4
    http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/index.html
  - Microsoft: JDBC tip 4
    http://msdn.microsoft.com/en-us/data/aa937724.aspx
  - MySQL: npr. Connector/J JDBC tip 4
    http://www.mysql.com/products/connector/
  - PostgreSQL: JDBC tip 4
    https://jdbc.postgresql.org/

Priprava sistema

- Inštaliramo in lociramo JDBC gonilnike:
  - PostgreSQL: potreben prenos gonilnika
  - MySQL: potreben prenos gonilnika Connector/J
  - Oracle: s produkti (client, sqldeveloper) ali ločeno
- Popravimo sistemsko spremenljivko CLASSPATH tako, da vključuje enega od omenjenih gonilnikov (odvisno od uporabljenega SUPB), ali pa gonilnike eksplicitno vključimo v projekt

Predstavitev preprostega JDBC programa



Pisanje JDBC programa

1. Napovej uporabo potrebnih razredov
2. Naloži JDBC gonilnik s pomočjo objekta DriverManager
3. Identificiraj vir podatkov (data source)
4. Kreiraj objekt za povezavo s PB (Connection).
5. Kreiraj objekt, ki predstavlja SQL stavek (Statement).
6. Izvedi poizvedbo z uporabo Statement objekta.
7. Množico rezultatov preberi iz vrnjenega ResultSet objekta.
8. Zapri objekt ResultSet.
9. Zapri objekt Statement.
10. Zapri objekt Connection.

Napoved uporabe potrebnih razredov

```
import java.sql.*;
```

ali bolj eksplicitno

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

Nalaganje JDBC gonilnika

- Za vsak SUPB ima gonilnik svoje ime, s katerim ga naložimo
- Za PostgreSQL:
  - Class.forName("org.postgresql.Driver");
- Za MySQL:
  - Class.forName("com.mysql.jdbc.Driver");
- JDBC od 4.0 dalje se avtomatsko naloži potrebne gonilnike iz "classpath", eksplicitno nalaganje ni potrebno (pravi gonilnik se avtomatsko izbere glede na povezovalni niz)

Identifikacija vira podatkov

- Specifikacija s povezovalnim nizom
  - Zgradba niza:
    jdbc:protokol:identifikacija vira podatkov
- Za PostgreSQL:
  - sourceURL = "jdbc:postgresql://racunalnik:port/baza";
  - Običajna številka porta je 5432
- MySQL:
  - sourceURL = "jdbc:mysql://racunalnik:port/shema";
  - Običajna številka porta je 3306
- Za Oracle:
  - sourceURL = "jdbc:oracle:thin:@racunalnik:port:baza";
  - Običajna številka porta je 1521

Kreiranje povezave s PB

- Preko DriverManagerja:
  Connection databaseConnection =
  DriverManager.getConnection(sourceURL, username, password);

- Preko DataSource vmesnika (z uporabo Java Naming and Directory Interface - JNDI)
  - Simbolno poimenovanje, npr. preko XML, podobno kot DSN pri ODBC

```
<Resource
    name="jdbc/UsersDB"
    auth="Container"
    type="javax.sql.DataSource"
    maxActive="100"
    maxIdle="30"
    maxWait="10000"
    driverClassName="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost:3306/usersDB"
    username="database_user"
    password="secret_password"
/>
```

```
// … create context …

DataSource ds = (DataSource)
                envContext.lookup("jdbc/UsersDB");

Connection databaseConnection = ds.getConnection();
```

Kreirajnje in izvajanje SQL stavka

- Statement statement =
  databaseConnection.createStatement();
- ResultSet result = statement.executeQuery("SELECT …");

- Obstajajo še druge vrste stavkov
  - PreparedStatement (vnaprej pripravljena poizvedba, primerno za večkratno izvajanje s posredovanjem vhodnih – IN - parametrov)
  - CallableStatement (klici shranjenih procedur v bazi s posredovanjem vhodnih – IN – in izhodnih – OUT - parametrov)

Obdelava množice rezultatov

1. Izpis meta-podatkov (imena stolpcev)

```
ResultSetMetaData md = result.getMetaData();
for (int i=1; i<=md.getColumnCount(); i++) {
    System.out.print(md.getColumnLabel(i)+"\t");
 }
 System.out.println();
```

2. Izpis dejanskih rezultatov (vrednosti atributov)

```
 while (result.next()) {
    for (int i=1; i<=md.getColumnCount(); i++) {
          System.out.print(result.getString(i)+"\t");
    }
    System.out.println();
}
```

Delo z rezultati (ResultSet)

- ResultSet result = statement.executeQuery("SELECT …");
- Prehajanje med vrsticami rezultata:
  - Metodi first() in last ()
  - Metodi previous() in next(): prehod na predhodno/naslednjo vrstico

- Metode za dostop do vrednosti atributov, npr:
  - getString("ime atributa”) ali getString(pozicija atributa)
  - Različne metode (glede na vnaprej znan tip atributa)

`getAsciiStream()`	`getTimestamp()`	`getTime()`
`getBoolean()`	`getBinaryStream()`	`getString()`
`getDate()`	`getBytes()`	`getByte()`
`getInt()`	`getFloat()`	`getDouble()`
`getShort()`	`getObject()`	`getLong()`

Preslikava podatkovnih tipov

Java Object/Type	JDBC Type
Int	INTEGER
Short	SMALLINT
Byte	TINYINT
Long	BIGINT
Float	REAL
Double	DOUBLE
java.math.BigDecimal	NUMERIC
Boolean	BOOLEAN or BIT
String	CHAR, VARCHAR, or LONGVARCHAR
Clob	CLOB
Blob	BLOB
Struct	STRUCT
Ref	REF
java.sql.Date	DATE
java.sql.Time	TIME
java.sql.Timestamp	TIMESTAMP
java.net.URL	DATALINK
Array	ARRAY
byte[]	BINARY, VARBINARY, or LONGVARBINARY
Java class	JAVA_OBJECT

Obravnava napak

- Class.forName("Ime gonilnika") ali avtomatsko nalaganje gonilnika ob napaki vrže ClassNotFoundException
- Ostale JDBC metode ob napaki vržejo SQLException
- Zato moramo te napake obravnavati:

```
try {
.... // JDBC klici
} catch(ClassNotFoundException cnfe) {
    System.err.println(cnfe);
} catch(SQLException sqle) {
    System.err.println(sqle);
}
```

Primer dostopa do podatkov

```
import java.sql.*;

public class TestJDBC1 {
   public static void main(String[] args) {
      try {
          String sourceURL;
          Connection databaseConnection;
          if (!true) { // Oracle
              Class.forName("oracle.jdbc.driver.OracleDriver");
              //  Create a connection through the DriverManager
              sourceURL =
                  "jdbc:oracle:thin:@todo.fri.uni-lj.si:1521:vaje";
          } else { // MySQL
              Class.forName("com.mysql.jdbc.Driver");
              //  Create a connection through the DriverManager
              sourceURL = "jdbc:mysql://localhost:3306/vaje";
          }
          databaseConnection =
                  DriverManager.getConnection
                    (sourceURL, "pb", "pbvaje");
          System.out.println("Connection is: "+databaseConnection);
```

Primer dostopa do podatkov

```
                    Statement statement = DatabaseConnection.createStatement();
                    ResultSet result = statement.executeQuery
                                       ("SELECT * FROM jadralec");
                    // Meta podatki
                    ResultSetMetaData md = result.getMetaData();
                    int count = md.getColumnCount();
                    for (int i=1; i<=count; i++)
                        System.out.print(md.getColumnLabel(i)+"\t");
                     System.out.println();
                     // Izpis v tekstovni obliki
                     while (result.next()) {
                          for (int i=1; i<=count; i++)
                               System.out.print(result.getString(i)+"\t");
                          System.out.println();
                        }
                    // Zapri ResultSet , Statement, Connection
                    result.close(); statement.close(); databaseConnection.close();
            } catch(ClassNotFoundException cnfe) {
                    System.err.println(cnfe);
            } catch(SQLException sqle) {
                    System.err.println(sqle);
            }
        }
    }
```

Kje lahko uporabljamo JDBC

- Poljubna aplikacija (JApplication) v Javi
  - Ob uporabi gonilnikov tretje ali četre skupine (čista Java) so aplikacije poljubno prenosljive
- Poljubni apleti (JApplet) ali servleti (JSP) (podprto do Java SE 8)
  - Ob uporabi gonilnikov tretje ali četre skupine (čista Java) so apleti/servleti poljubno prenosljive
  - Možnost pisanja popolnoma prenosljivih spletnih aplikacij, ki dostopajo do PB
- Mobilne aplikacije (Android)
  - Uporaba je mogoča
  - Zaradi varnosti ni priporočljiva (boljše dostop preko spletnih servisov)
- Spletne aplikacije (JavaScript) ???