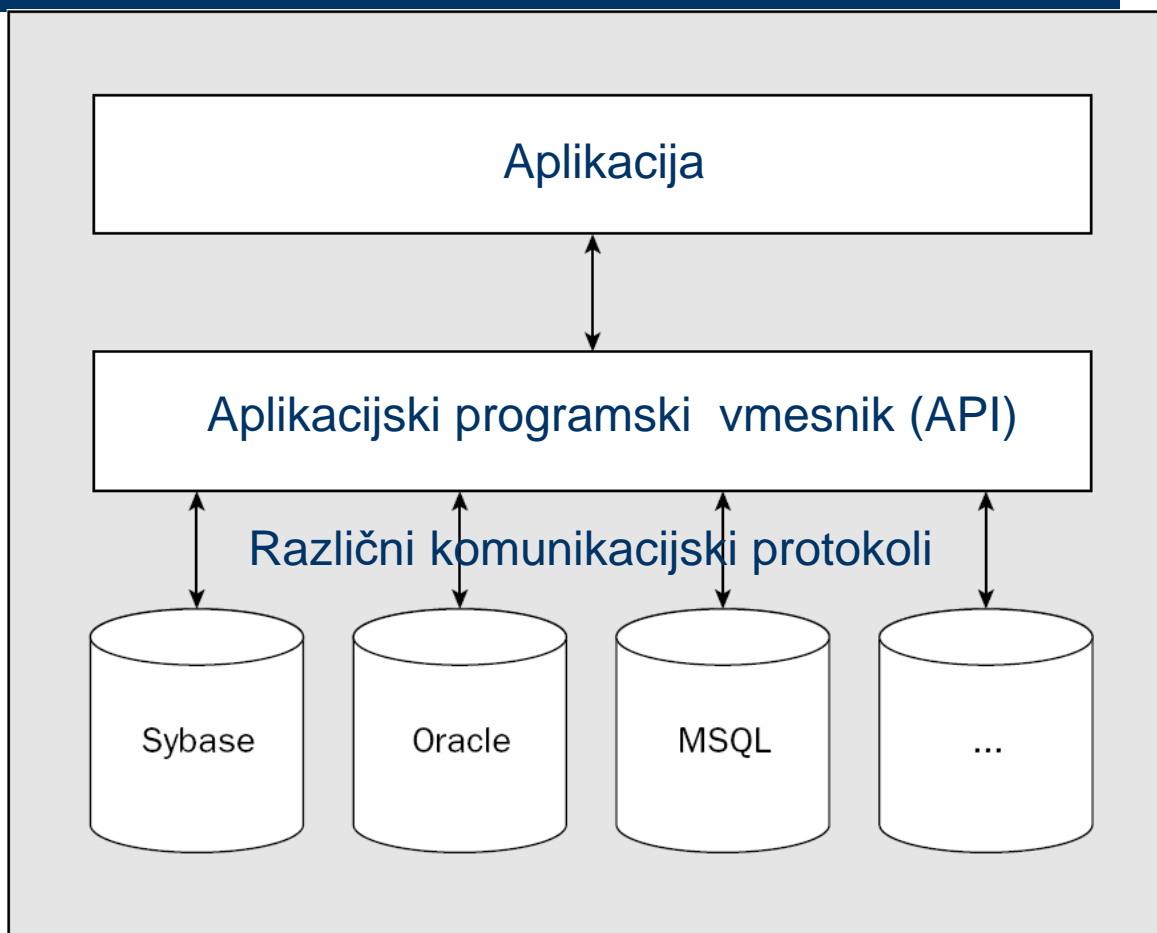


Komunikacija med aplikacijo in SUPB

Podatkovna
baza



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 so tako 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 Microsofta s podjetjem Simba Technologies
- Sloni na različnih standardnih Call Level Interface (CLI) specifikacijah iz SQL Access Group, X/Open in ISO/IEC
- 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).

ODBC

- Prevzeli so ga vsi pomembnejši proizvajalci SUPB
- Množica implementacij ODBC gonilniških sistemov za različne operacijske sisteme in SUPB-je:
 - Microsoft ODBC (DAO, DAC: data access objects, data access components),
 - iODBC (open source: MacOS, Linux, Solaris, ...),
 - IBM i5/OS (IBM, DB2),
 - UnixODBC (open source: Linux),
 - UDBC (predhodnik iODBC, združuje ODBC in SQL access group CLI)
 - Oracle, Informix, Sybase, MySQL, ... za različne OS

Značilnosti ODBC

- Proceduralni programski vmesnik za dostop do podatkovne baze
- 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 naš OS in ODBC gonilnik za naš OS in uporabljano PB

Zakaj ODBC?

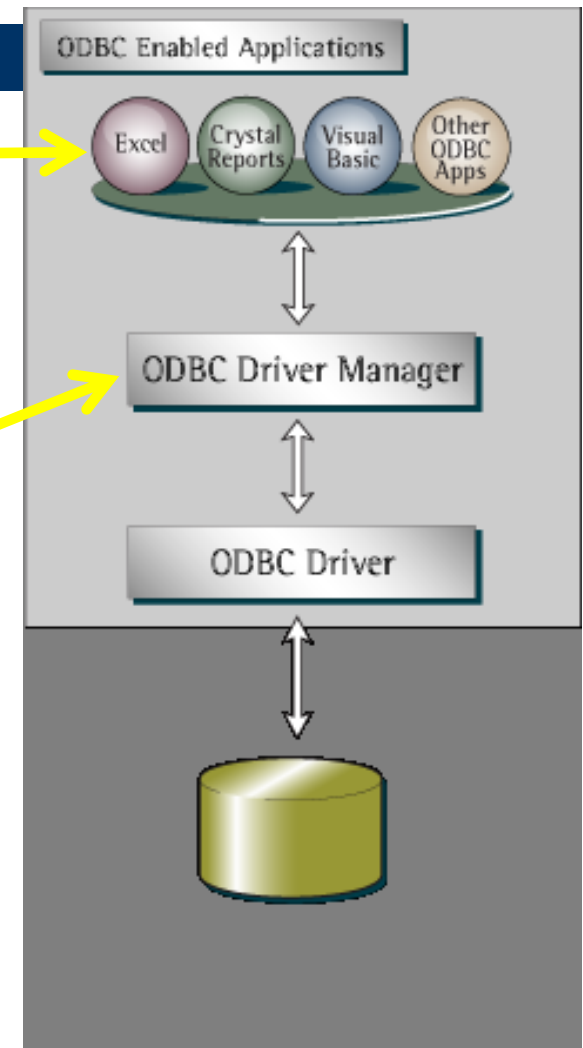
- Aplikacije niso vezane na konkreten API
- SQL stavke lahko v kodo vključujemo statično ali dinamično
- Aplikacij ne zanima dejanski komunikacijski protokol
- Format podatkov prilagojen programskemu jeziku
- Standardiziran vmesnik (X/Open, ISO CLI)
- Univerzalno sprejet in podprt

Kaj nam ODBC ponuja

- Gonilnike, ki omogočajo poenoten dostop do PB
- 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

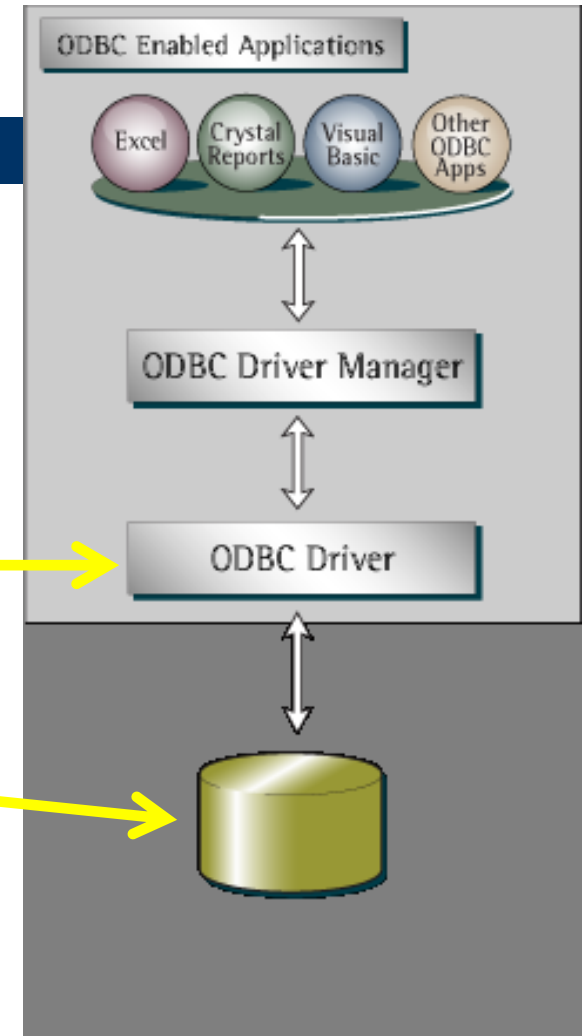
Arhitektura ODBC

- 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

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

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 operantorji (npr. SUM, MIN, ...)
 - Podatkovni tipo: 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 podatkovno bazo
- pyodbc je odprtokoden, uporablja MIT licenco, in ga lahko zastonj uporabljamo tako v pridobitne, kot nepridobitne namene (vključno z izvorno kodo)
- domača stran in dokumentacija:

<http://code.google.com/p/pyodbc/>

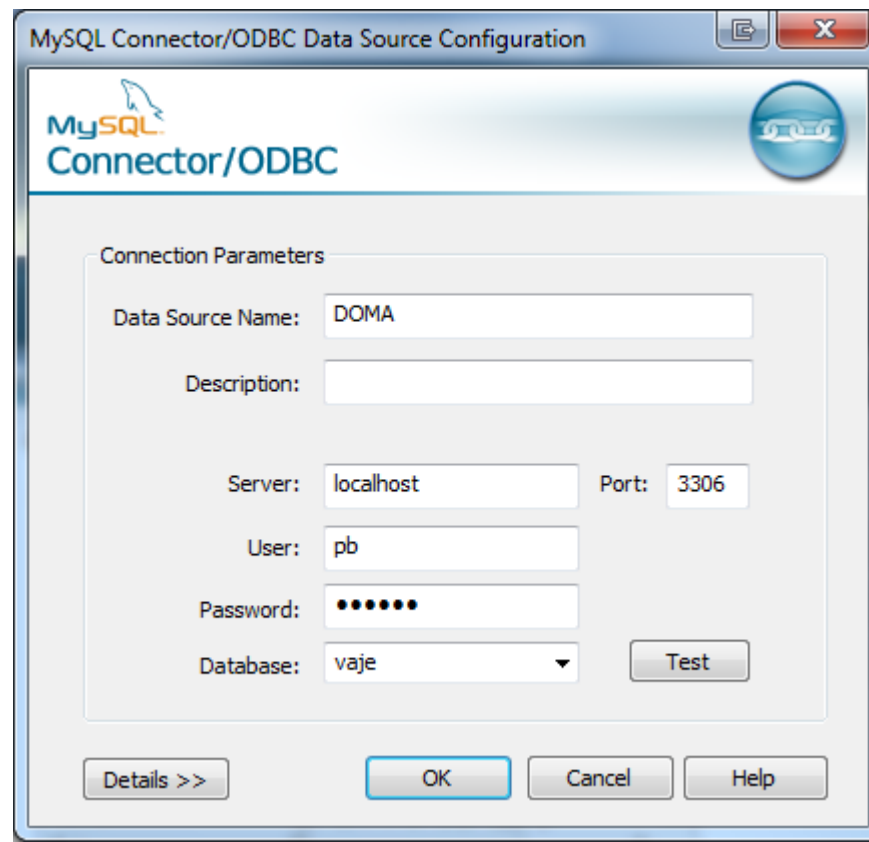
Poglejte si!!!

Predpriprava na uporabo pyodbc

- SUPB s podatki
- Python in pyodbc (za pripadajočo verzijo Pythona)
 - <https://code.google.com/p/pyodbc/>
- Ali pypyodbc (1.3.5) namesto pyodbc
 - <https://pypi.org/project/pypyodbc/>
- Delovno okolje: Pythonwin, Idle, Pycharm
- ODBC gonilnik za izbrani OS in SUPB
 - <https://dev.mysql.com/downloads/connector/odbc/> ali
 - MySQL: Connector/ODBC (na učilnici: instalirajte)

Priprava podatkovnega vira (MySQL)

- Odprite Control Panel->Administrative tools
->Set up ODBC data sources
- V zavihku User DSN izberite Add in nato določite ODBC gonilnik:
 - MySQL ODBC 8.0 ANSI driver
 - Prepišite vrednosti z desne slike: DSN je lahko poljuben.
 - Lahko vnesete uporab. ime in geslo



The screenshot shows the 'MySQL Connector/ODBC Data Source Configuration' window. It contains the following fields and controls:

- Connection Parameters** section:
 - Data Source Name:** Text box containing 'DOMA'.
 - Description:** Empty text box.
 - Server:** Text box containing 'localhost'.
 - Port:** Text box containing '3306'.
 - User:** Text box containing 'pb'.
 - Password:** Text box with masked characters (dots).
 - Database:** Dropdown menu showing 'vaje'.
 - Test** button.
- At the bottom: **Details >>** button, **OK** button, **Cancel** button, and **Help** button.

Osnovni gradniki pyodbc

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

pyodbc: povezava

- Povezavo ustvarimo z ukazom:
`c = pyodbc.connect(ConnectionString)`
- ConnectionString določa povezavo, npr.
ConnectionString = 'DSN=FRI;UID=pb;PWD=pbvaje'
ali
ConnectionString = 'DSN=DOMA;UID=pb;PWD=pbvaje'

pyodbc: povezava

- Za MySQL bi ConnectionString lahko napisali tudi brez definiranega DSN, npr.:

```
ConnectionString = 'DRIVER={MySQL ODBC 8.0 ANSI Driver};  
SERVER=pb.fri.uni-lj.si;  
DATABASE=vaje;  
UID=pb;  
PWD=pbvaje;'
```

pyodbc: povezava

- Nekatere metode:
 - close(): zapri povezavo, enako pri destruktorku objekta
 - commit(): uveljavi transakcijo (če SUPB podpira)
 - rollback(): razveljavi transakcijo (če SUPB podpira)
 - cursor(): vrne nov kurzorski objekt, ki uporablja povezavo
- Kurzorji: izvajajo SQL stavke in omogočajo iteracijo po vrsticah rezultata

pyodbc: kurzor

- Kurzor x ustvarimo z ukazom:
`x = c.cursor()` # c je povezava
- Nekateri atributi:
 - description: opis stolpcev rezultata (shema)
 - rowcount: število vrstic rezultata
- Nekaterne metode:
 - execute(ukaz, [parametri]): izvede ukaz z opsijskimi 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 iteriramo po v
- Parametri v SQL ukazih:
 - Primer: **SELECT * FROM** jadralec
 - SQL ukaz kot niz znakov: Pythonov način parametrizacije
 - x.execute('SELECT %s FROM %s' % ('*', 'jadralec'))
 - pyodbc prenos parametrov v metodi execute: (varneje)
 - ? označuje parameter
 - seznam parametrov 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 ( SQLKaz)
    ...
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

Primer programa

- Naloga: Postaraj jadralce za eno leto.
- Izvedba:
 - ustvari novo tabelo: postarani
 - v vsaki vrstici povečaj starost za 1
- Vse to lahko naredimo direktno v SQL-u. Kako? Vaja prejšnjega tedna.

Primer programa

```
import pyodbc
cnxn = pyodbc.connect('DSN=FRI;UID=pb;PWD=pbvaje')
cursor = cnxn.cursor()
updater= cnxn.cursor()
try:
    cursor.execute("DROP TABLE postarani")
except pyodbc.DatabaseError:
    pass
cursor.execute("CREATE TABLE postarani AS SELECT * FROM jadralec")
cnxn.commit()
```

**Izbriši, če že
obstaja tabela.**

Primer programa

```
cursor.execute("SELECT * FROM postarani")
print("PRED")
for r in cursor:
    print(r)
cursor.execute("SELECT * FROM postarani")
for r in cursor:
    updater.execute("UPDATE postarani SET starost =? WHERE jid =?",
                    r.starost + 1, r.jid)
cursor.execute("SELECT * FROM postarani")
print("PO")
for r in cursor:
    print(r)
cnxn.commit()
```

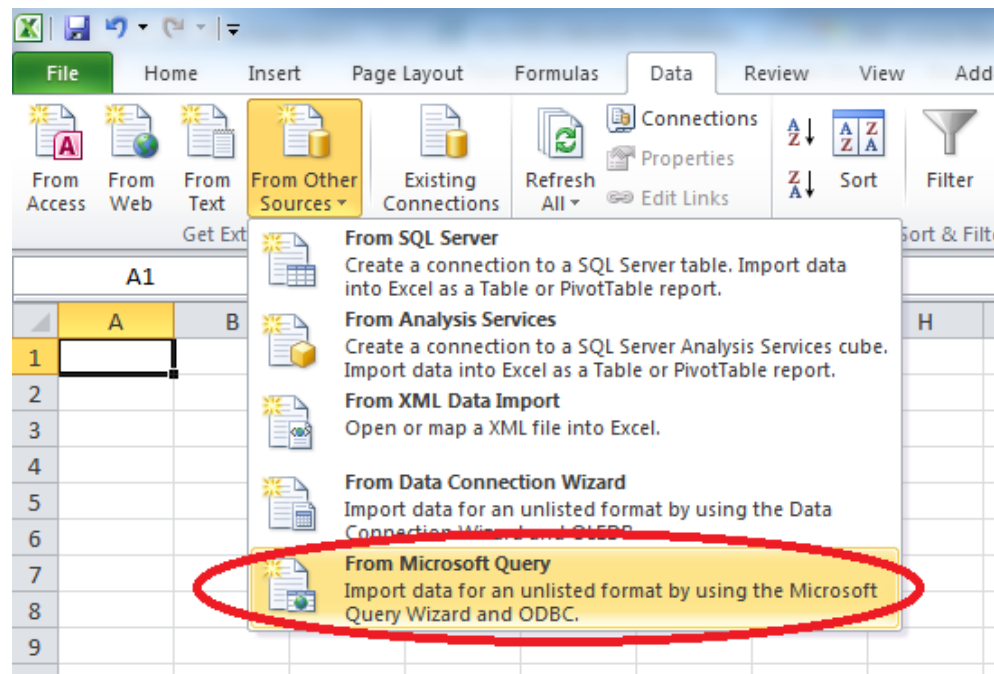
**Zakaj še en
SELECT?**

**Pazite na
velikost črk
pri imenih
atributov**

**COMMIT zares
zapiše v bazo**

Microsoft Excel in ODBC

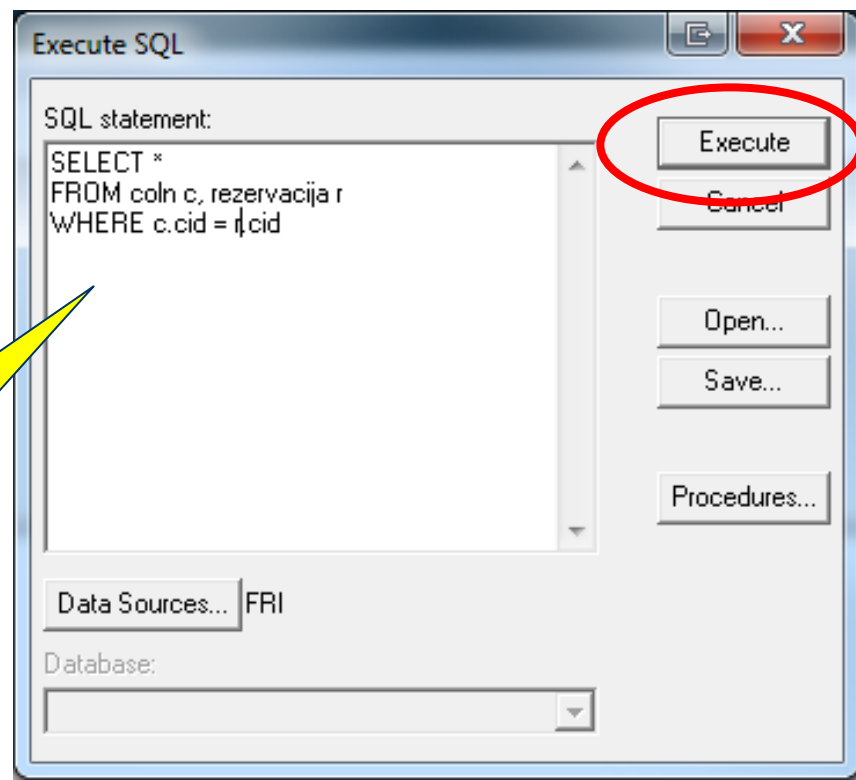
- Izberite
Data->From Other Sources->From Microsoft Query
- Izberite vir podatkov, kot definirano v ODBC Data Sources (npr. FRI)



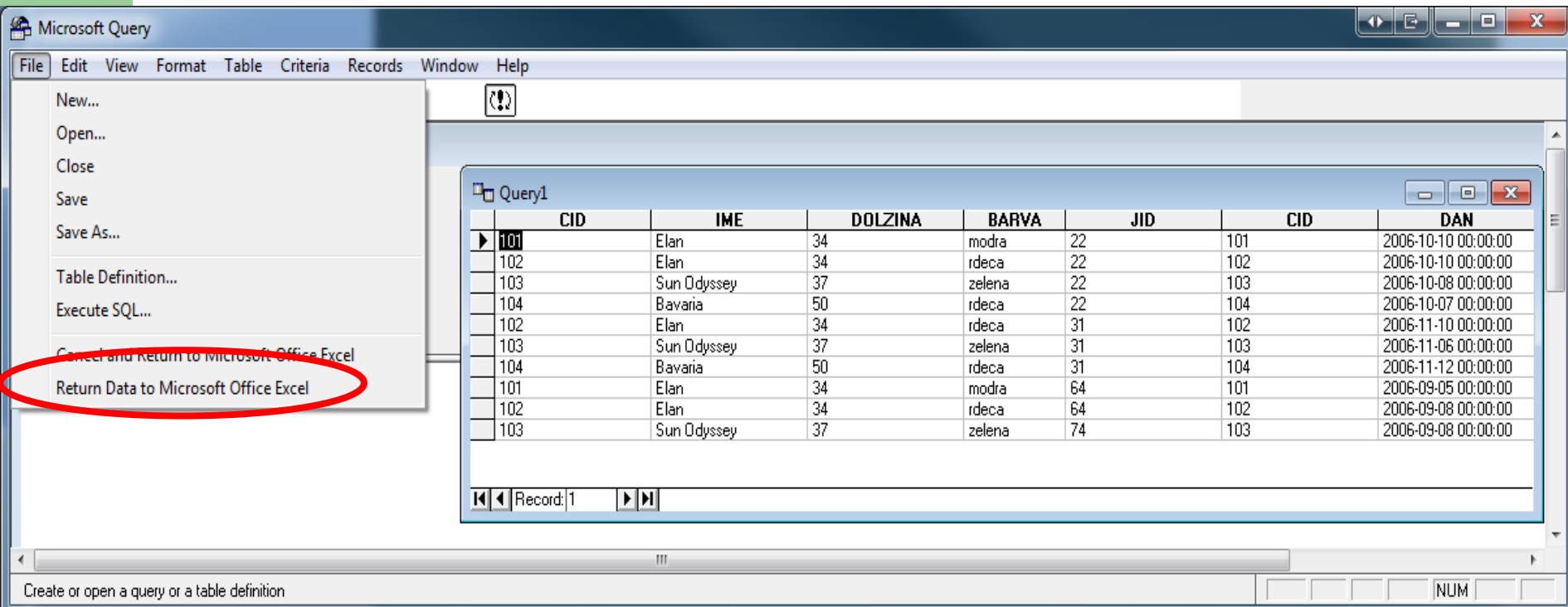
Microsoft Excel in ODBC

- 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.

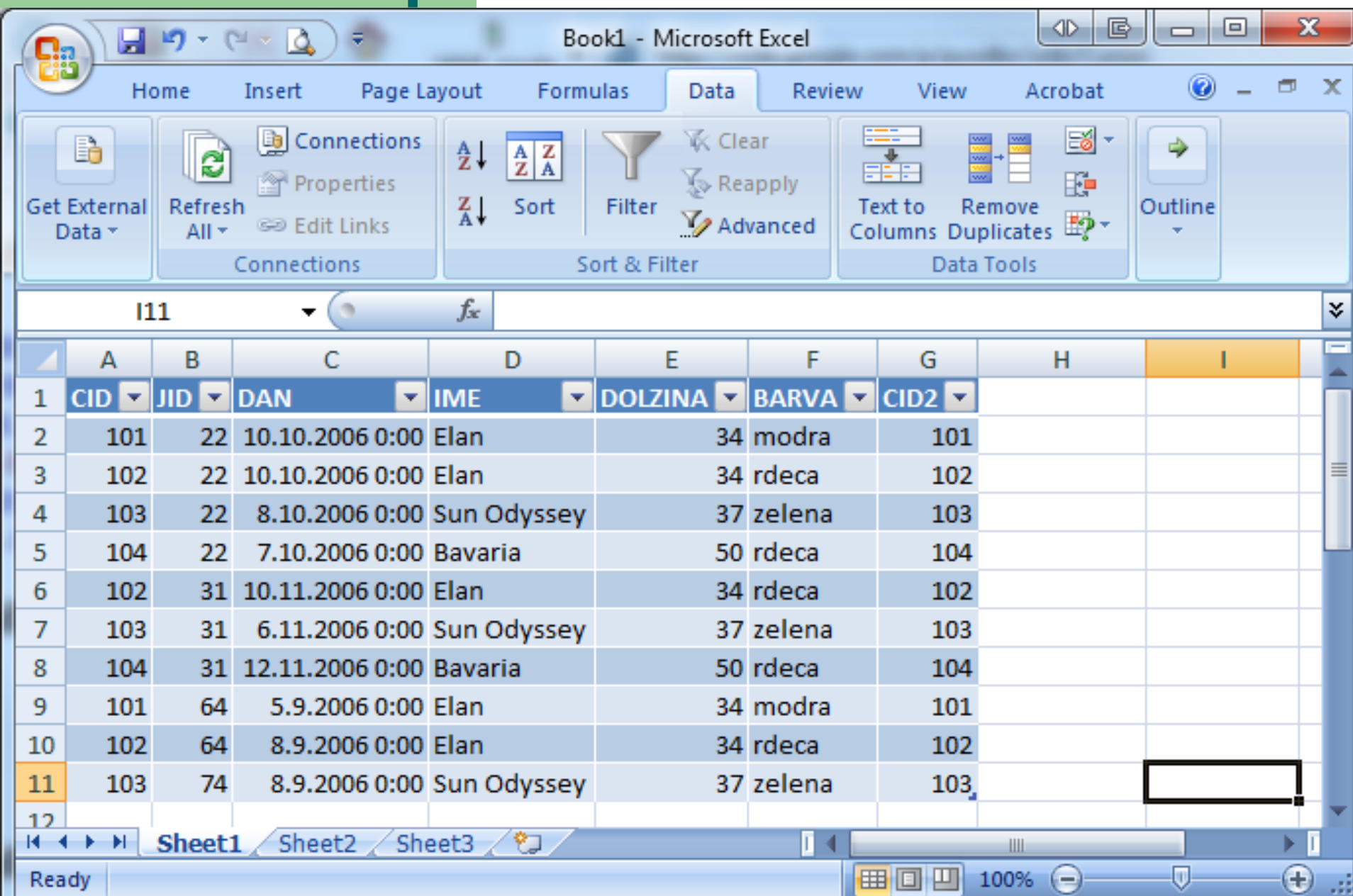


Microsoft Excel in ODBC



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

Rezultat poizvedbe v Excelu



Book1 - Microsoft Excel

Home Insert Page Layout Formulas Data Review View Acrobat

Get External Data Refresh All Connections Properties Edit Links Sort & Filter Sort Filter Clear Reapply Advanced Data Tools Text to Columns Remove Duplicates Outline

11 fx

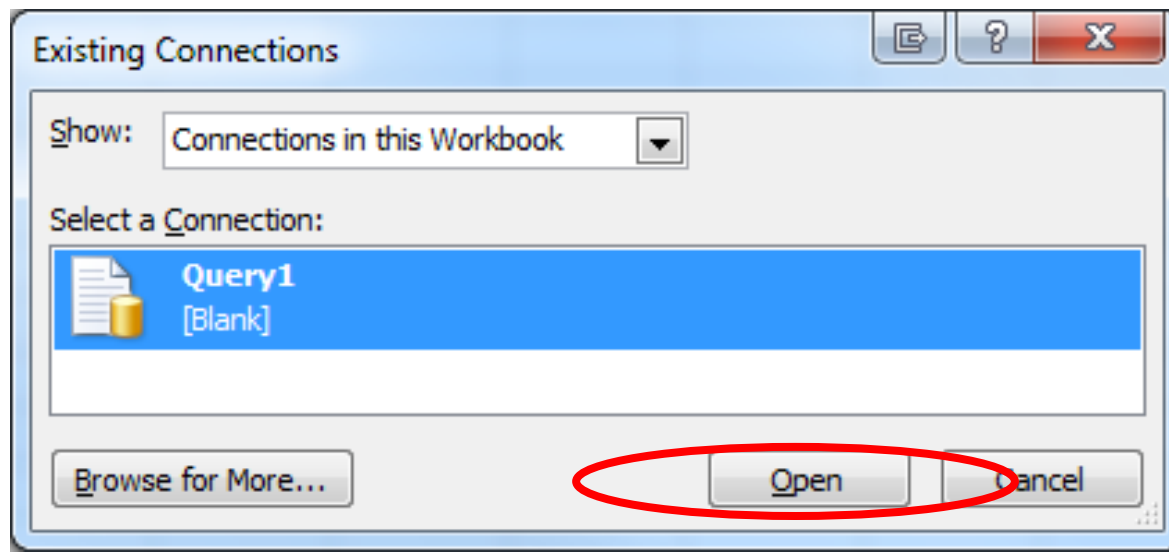
	A	B	C	D	E	F	G	H	I
1	CID	JID	DAN	IME	DOLZINA	BARVA	CID2		
2	101	22	10.10.2006 0:00	Elan	34	modra	101		
3	102	22	10.10.2006 0:00	Elan	34	rdeca	102		
4	103	22	8.10.2006 0:00	Sun Odyssey	37	zelena	103		
5	104	22	7.10.2006 0:00	Bavaria	50	rdeca	104		
6	102	31	10.11.2006 0:00	Elan	34	rdeca	102		
7	103	31	6.11.2006 0:00	Sun Odyssey	37	zelena	103		
8	104	31	12.11.2006 0:00	Bavaria	50	rdeca	104		
9	101	64	5.9.2006 0:00	Elan	34	modra	101		
10	102	64	8.9.2006 0:00	Elan	34	rdeca	102		
11	103	74	8.9.2006 0:00	Sun Odyssey	37	zelena	103		
12									

Sheet1 Sheet2 Sheet3

Ready 100%

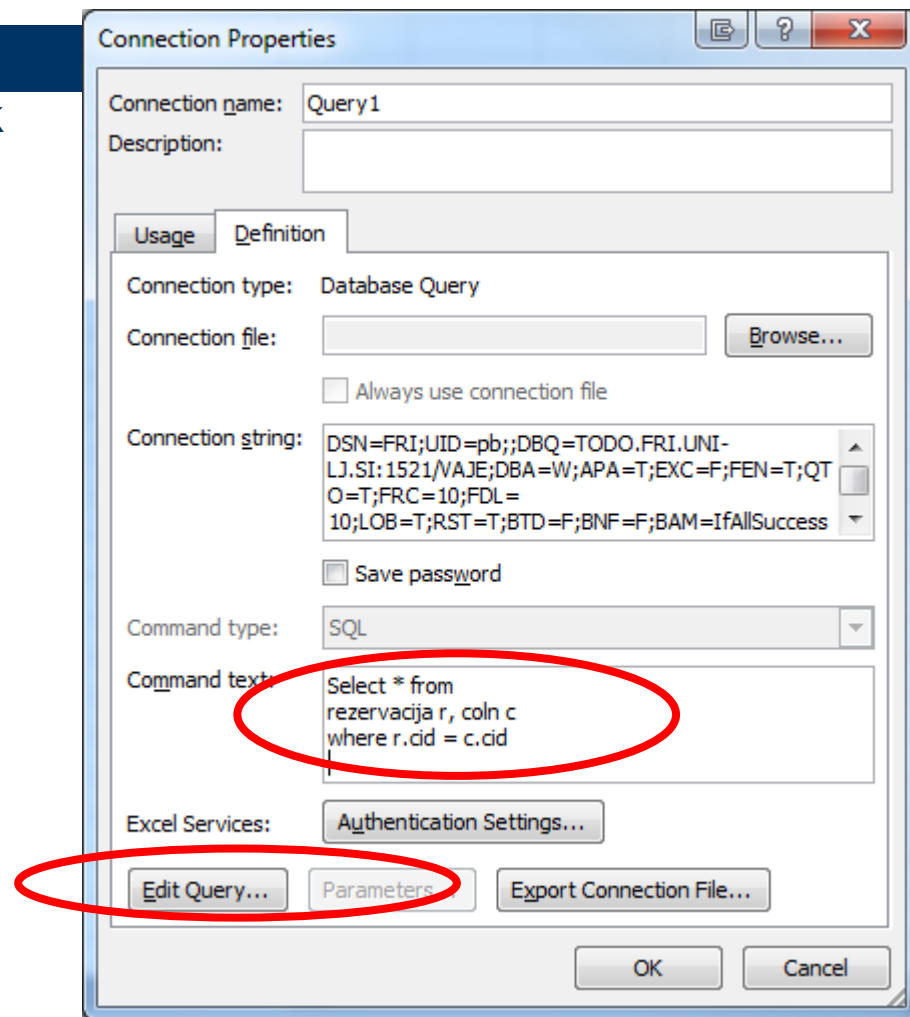
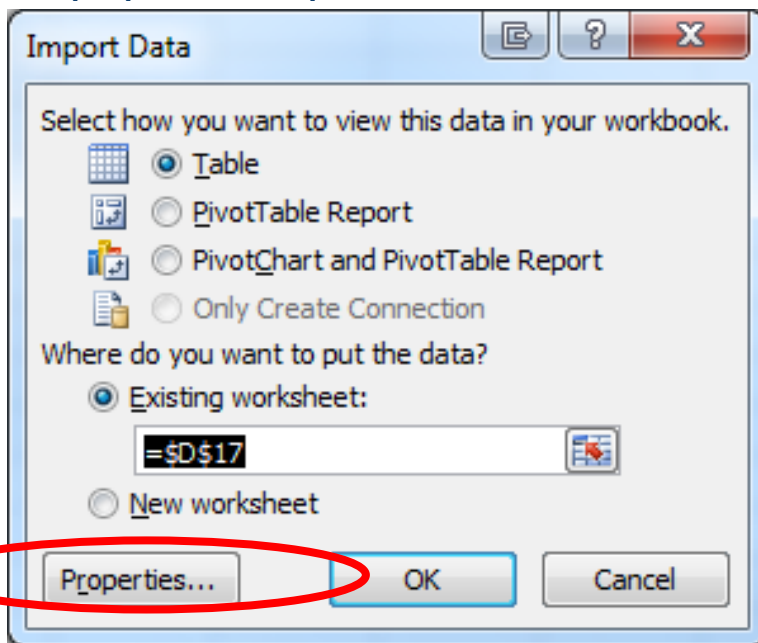
Microsoft Excel in ODBC

- Povezave s podatkovno bazo so "žive"
 - S pritiskom na Data-> Refresh All osvežite vsebino rezultata poizvedbe
- 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



SQLite in Python

- SQLite uporablja datotečni sistem ali pomnilnik za shranjevanje podatkov.
- Dobite ga na strani www.sqlite.org
- V Pythonu je modul sqlite3 že vključen.

Primer povezave z sqlite:

```
import sqlite3
#Ustvari povezavo na datoteko z imenom PBBaza.db
c = sqlite3.connect('PBBaza.db')
#Od tukaj naprej je enako kot uporaba pyodbc
x = c.cursor()
x.execute(SQLstavki)
c.close()
```

SQLite in Python

Naredite lahko tudi povezavo na pomnilnik, kar pomeni, da se bo baza po zaprtju programa izgubila.

```
import sqlite3  
c = sqlite3.connect(':memory:')  
#vaš program  
c.close()
```

SQLite (sqlite3.exe)

Bazo SQLite lahko uporabljate tudi samostojno. Iz uradne strani www.sqlite.org prenesite sqlite3.exe, ki ga dobite v arhivu [sqlite-tools-win32-x86-3270200.zip](#).

Povezava na bazo: sqlite3 PBBaza.db
Baza mora biti v isti mapi, kot sqlite3.

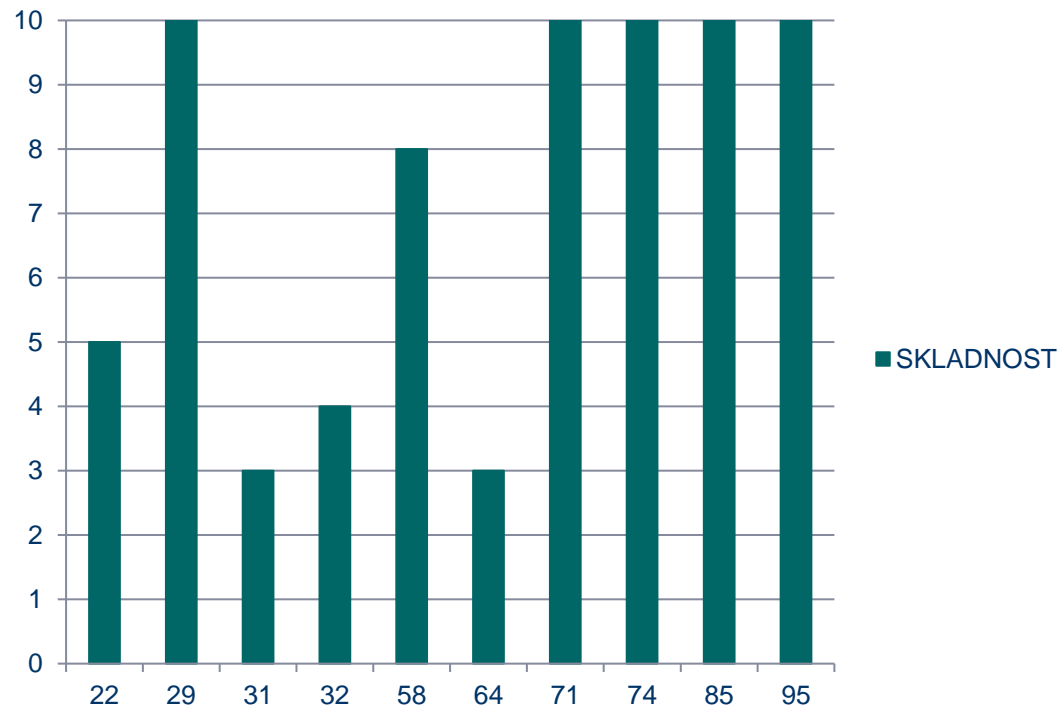
Lahko tudi poženete: sqlite3
Ter nato znotraj SQLite napišete: .open PBBaza.db

Poženete lahko tudi skripto iz diska: .read jadralci.sql

NALOGA V EXCELU

- Povežite se na bazo in prenesite tabelo optimal
- Narišite graf te tabele, kot ga kaže slika.

JID	CID	SKLADNOST
22	104	5
29	102	10
31	104	3
32	104	4
58	104	8
64	104	3
71	104	10
74	101	10
85	101	10
95	102	10



NALOGA (Python - Employees)

Napišite funkcijo 'izpis(cursor, ids)', ki bo za vsak id v seznamu ids izpisala podatke o plačah čimbolj podobno kot vidite na spodnji sliki. Za vsakega zaposlenega se izpiše le 5 najvišjih izplačil, 2 najvišji pa sta obarvani zeleno. Rdeče so obarvane šifre vodji.

Za spremembo barve lahko uporabljate paket colorama.

```
from colorama import Fore
print("Navadno" + Fore.RED + "Rdeče" + Fore.RESET + "Navadno")
```

Id	Ime	Priimek	od	do	izplačilo
110022	Margareta	Markovitch	1997-12-29	1998-12-29	98843
110022	Margareta	Markovitch	1998-12-29	1999-12-29	100014
110022	Margareta	Markovitch	1999-12-29	2000-12-28	100592
110022	Margareta	Markovitch	2000-12-28	2001-12-28	104485
110022	Margareta	Markovitch	2001-12-28	9999-01-01	108407
10001	Georgi	Facello	1998-06-23	1999-06-23	81097
10001	Georgi	Facello	1999-06-23	2000-06-22	84917
10001	Georgi	Facello	2000-06-22	2001-06-22	85112
10001	Georgi	Facello	2001-06-22	2002-06-22	85097
10001	Georgi	Facello	2002-06-22	9999-01-01	88958
111877	Xiaobin	Spinelli	1997-08-15	1998-08-15	66104
111877	Xiaobin	Spinelli	1998-08-15	1999-08-15	66678
111877	Xiaobin	Spinelli	1999-08-15	2000-08-14	67470
111877	Xiaobin	Spinelli	2000-08-14	2001-08-14	70158
111877	Xiaobin	Spinelli	2001-08-14	9999-01-01	71406

NALOGA (Python)

- V Pythonu napišite program, ki za vsakega jadralca izpiše šifro najbolj “*skladnega*” čolna.
- Rezultate zapišite v tabelo `optimal(jid, cid, skladnost)`
- Najbolj “*skladen*” čoln *cid* za jadralca *jid* je tisti, ki po formuli $(jid + cid) \% 11$ daje najvišjo vrednost.

NALOGA V SQLite

- Ustvarite svojo bazo na disku in jo napolnite s podatki iz skripte jadralci.sql. Uporabite sqlite3.exe.
 - Pozor: Skripto predelajte tako, da ne bo vračala napak, saj je sintaksa SQLite nekoliko drugačna od sintakse MariaDB.
 - Na narejeno bazo se povežite s Pythonom in izpišite vse tabele. Preverite, da so podatki v tabelah pravilni.