

# LABORATORY OF DATA SCIENCE

## Data Access: Relational Data Bases

# RDBMS data access

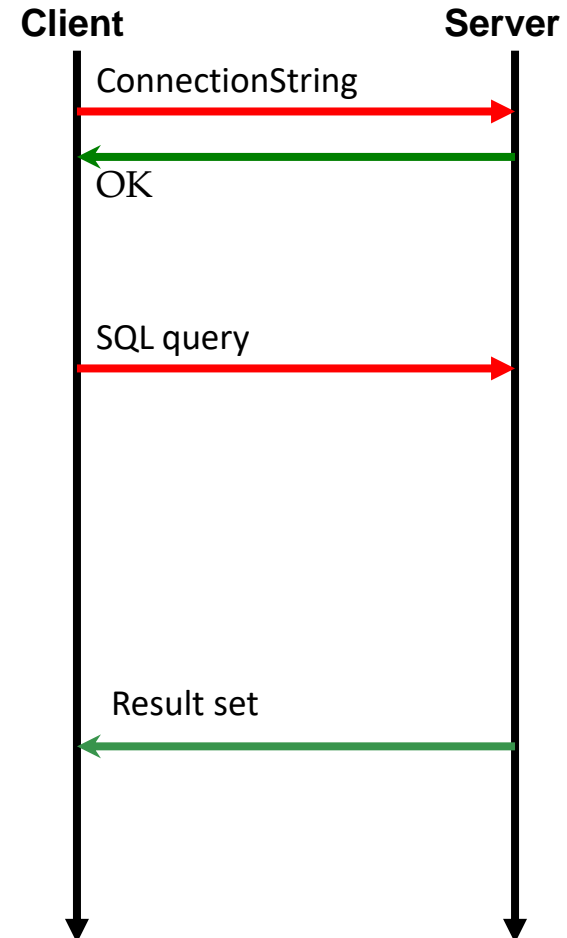
2

- Protocols and API
  - ▣ ODBC, OLE DB, ADO, ADO.NET, JDBC
- Python DBAPI with ODBC protocol

# Connecting to a RDBMS

3

- **Connection protocol**
  - locate the RDBMS server
  - open a connection
  - user authentication
  
- **Querying**
  - query SQL
    - SELECT
    - UPDATE/INSERT/CREATE
  - stored procedures
  - prepared query SQL
  
- **Scan Result set**
  - scan row by row
  - access result meta-data



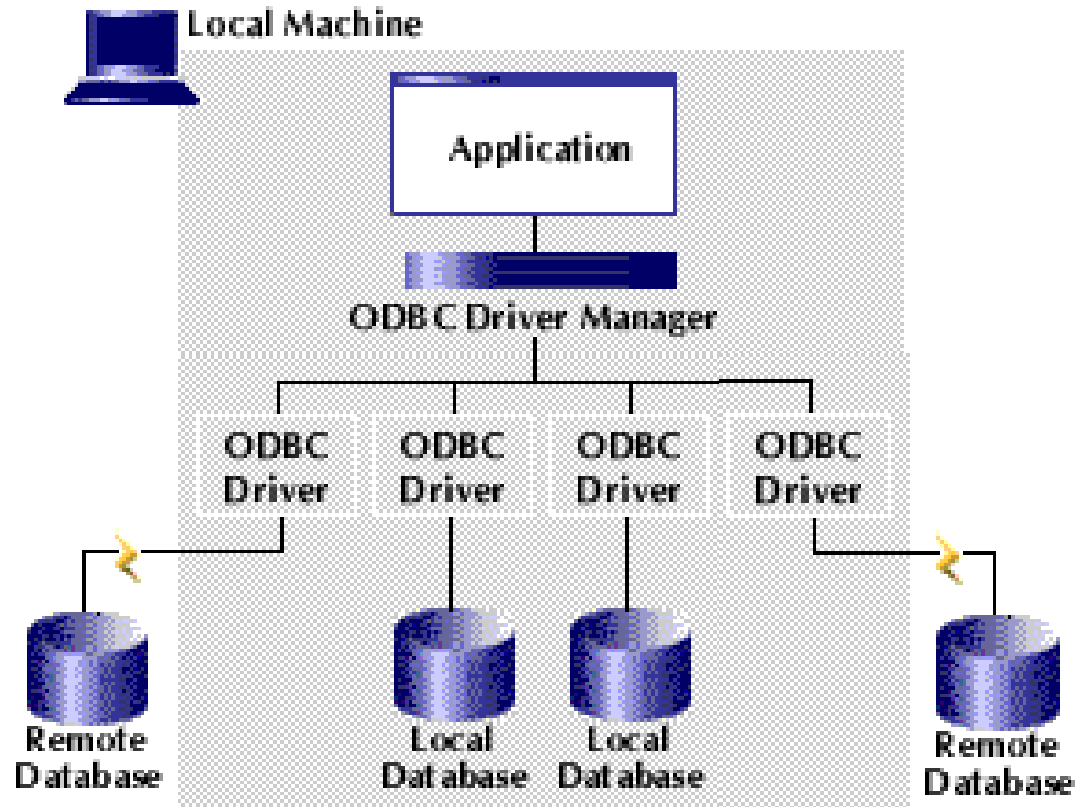
# Connection Standards

4

- ODBC - Open DataBase Connectivity
  - ▣ Windows: [odbc](#) Linux: [unixodbc](#), [iodbc](#)
  - ▣ Tabular Data
  
- JDBC – Java DataBase Connectivity
  
- [OLE DB](#) (Microsoft) – Object Linking and Embedding
  - ▣ Tabular data, XML, multi-dimensional data
  
- [ADO](#) (Microsoft) – ActiveX Data Objects
  - ▣ Object-oriented API on top of OLE DB
  
- [ADO.NET](#)
  - evolution of ADO in the .NET framework

# ODBC Open DataBase Connectivity

5



# ODBC Demo

6

- Registering an ODBC data source
  - ▣ pubs on access
  - ▣ pubs on SQL Server (driver SQL Server)
- Data access
  - ▣ copy Access table to Excel
- Linked tables
  - ▣ Linking SQL Server Table from Access

# OLE DB Demo

7

- Creating .udl data links
- Data access
  - ▣ accessing Access data from Excel
- Linked tables
  - ▣ accessing Excel data from Access
- OLE DB Drivers
  - ▣ By Microsoft

# RDBMS data access

8

- **Python DBAPI** is a standard specification for modules that interfaces with databases
- Most of Python database interfaces adhere to this standard
- **Functions:**
  - ▣ Connecting to a database
  - ▣ Submitting SQL queries
  - ▣ Scanning the results of queries
  - ▣ Accessing meta-data on tables



# Support of Different RDBMS

9

- **Portable across several relational and non-relational databases:**
  - Microsoft SQL Server
  - Oracle
  - MySQL
  - IBM DB2
  - PostgreSQL
  - Firebird (and Interbase)
  - Cassandra
  - MongoDB
  - .....

# Different Modules for a DB

10

Given a database we have various module options. For example, MySQL has the following interface modules:

- ❑ MySQL for Python (`import MySQLdb`)
- ❑ PyMySQL (`import pymysql`)
- ❑ pyODBC (`import pyodbc`)
- ❑ MySQL Connector/Python (`import mysql.connector`)
- ❑ mypysql (`import mypysql`)
- ❑ etc ...

# DBAPI Specification

11

- Most of database modules conform to the specification
- no matter which kind of database and/or module you choose, the code will likely look very similar
- See details here:  
<https://www.python.org/dev/peps/pep-0249/>

# DBAPI Specification

12

- Each module interface is required to have the following functions
- `connect (args)` : a constructor for **Connection objects**, that makes the access available. **Arguments** are **database-dependent**
- `conn.close ()` – close connection
- `conn.commit ()` – commit pending transaction
- ....

# DBAPI Specification

13

- `conn.cursor()` – return a Cursor object for the connection. Cursors are used fetch operations
- `c.execute(op, [params])` –prepare and execute an operation with parameters where the second argument may be a list of parameter sequences
- `c.fetch[one|many|all]([s])` – fetch next row, next *s* rows, or all remaining rows of result set
- `c.close()` – close cursor.
- and others.

# Programming pattern

14

1. Import the DB module
2. Connect to the RDBMS
3. Submit a SQL query
4. Process query results
5. Close the connection

# DB Module: Pyodbc

15

- ❑ Pyodbc is an **open source** Python module ODBC and implementing the DBAPI 2.0 specification.
- ❑ Enables an easily connection of Python applications to data sources with an ODBC driver
- ❑ Python program along with the pyodbc module will use an **ODBC driver manager** and **ODBC driver**
- ❑ The ODBC driver manager is **platform-specific**
- ❑ The ODBC driver is **database-specific**
- ❑ The ODBC driver manager and driver will connect, typically over a network, to the database server.

# Connect to the RDBMS

16

- Access the database via the **connection object**
- Use connect constructor to create a **connection** with database

```
conn = pyodbc.connect(parameters...)
```

- Create cursor via the connection

```
cur = conn.cursor()
```

- Connect function requires the “connection string”
- The connection string depends on the driver



# Connection String

17

## □ The connection strings:

```
DRIVER=Driver name;          SERVER=hostname;  
DATABASE=DBname;           UID=user;  
PWD=password
```

## □ In Python:

```
conn = pyodbc.connect(  
    'DRIVER={ODBC Driver 17 for SQL Server};  
    SERVER=tcp:apa.di.unipi.it;  
    DATABASE=Foodmart;  
    UID=lbi;  
    PWD=pisa')
```

# ODBC DRIVER

18

- Microsoft have written and distributed multiple ODBC drivers for **SQL Server**:
  - {SQL Server} - released with SQL Server 2000
  - {SQL Native Client} - released with SQL Server 2005 (also known as version 9.0)
  - {SQL Server Native Client 10.0} - released with SQL Server 2008
  - .....

# ODBC DRIVER

19

- {SQL Server Native Client 11.0} - released with SQL Server 2012
- {ODBC Driver 11 for SQL Server} - supports SQL Server 2005 through 2014
- {ODBC Driver 13 for SQL Server} - supports SQL Server 2005 through 2016
- {ODBC Driver 13.1 for SQL Server} - supports SQL Server 2008 through 2016
- {ODBC Driver 17 for SQL Server} - supports SQL Server 2008 through 2017

# Submit a SQL query

20

## Select String:

```
query = "SELECT name, age FROM students"
```

## Submit the SQL query and get the result

```
cursor.execute(query)
```

## UPDATE String:

```
update = "UPDATE students SET age = age + 1";
```

```
cursor.execute(update )
```

```
Conn.commit()
```

# Scan query results

21

## FETCHALL:

```
cursor.execute("SELECT TOP 10 education, gender FROM  
customer")  
rows = cursor.fetchall() // all rows in memory!!!  
for row in rows:  
    print (row[0], row[1]) //access by index  
    print(row.gender, row.education) //access by name
```

## CURSOR AS ITERATOR:

```
cursor.execute("SELECT TOP 10 education, gender FROM  
customer;"):  
for row in cursor:  
    print(row.gender, row.education)
```

# Close the connection

22

...

```
// close the cursor
```

```
cursor.close();
```

```
// close connection to the database
```

```
conn.close();
```

...