CS571
Advanced Programming Techniques

SQLite

Objectives

- Relational Databases and Database Management Systems
- SQL
- SQLite
- Relational database programming from Java
- Relational database programming from Python

Database

- A database is a structured collection of data
  - Databases consist of tables
  - Each table has columns
  - Each column has a data type
  - Each table has rows
  - A key maybe defined to uniquely identify a row

Customer (cid: INT, name: TEXT, city:TEXT)

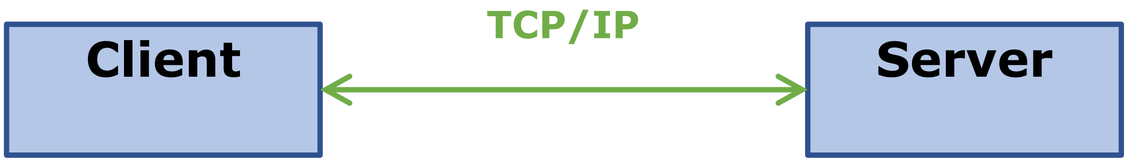
cid	name	city
1	Kevin Bacon	Philadephia
2	Kevin Hart	Philadelphia
3	Grace Kelly	Philadelphia

DBMS - Database Management System

- A DBMS is an system that…
  - Manages data centrally
  - Allows users to query the data
  - Enforces data constraints and other standards
  - Maintains data integrity and consistency
  - Applies security
  - Manages access to the data
  - Manages access to the data concurrently
  - Balances conflicting access to the same data
  - Provides high availability
  - And much more…

DBMS - Client-Server Architecture

- A relational DBMS follows the “Client-Server” Pattern
- The server is the DBMS
  - Listens on a known host
  - Listens on a given port
- The client is an application that connects to the DBMS
  - The clients connect to the server to perform read/write/updates operations on its data



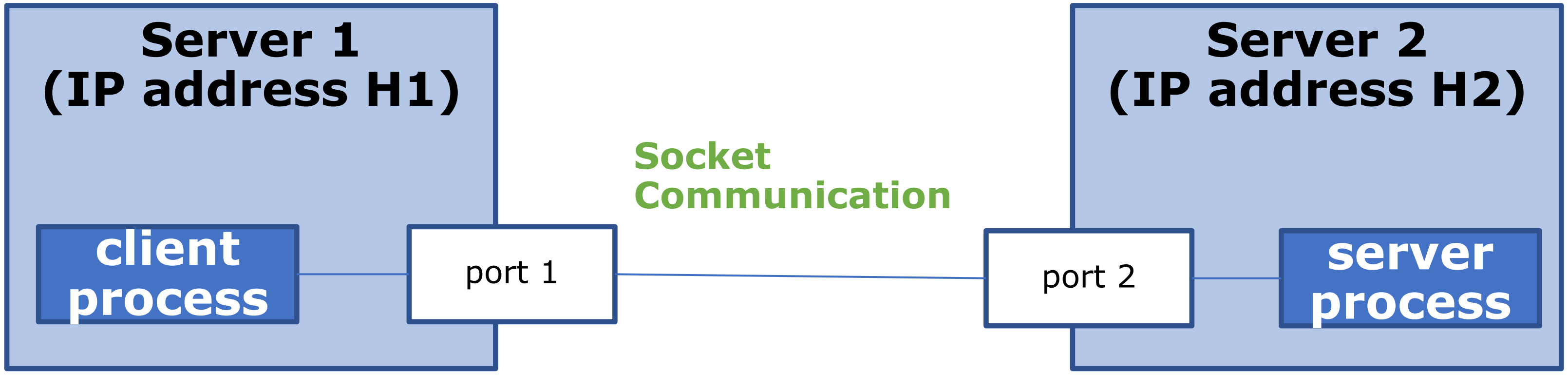
Network Address

- Computers operating in a network are assigned unique identifiers known as network addresses
- There are 3 flavors
- Media Access Control (MAC) Address
  - Assigned by manufacturers
  - Used by routers, Wi-Fi, Bluetooth, etc.
  - Example:30-65-EC-6F-C4-58
- Internet Address (IP)
  - Used by routers and people
  - Example 129.25.203.107
  - Example 127.0.0.1 (there is no place like 127.0.0.1)
- Domain Name (DN)
  - A DNS Server is used to convert DNS names to/form IP addresses
  - Example: tux.cs.drexel.edu
  - Example: localhost

Port and Socket

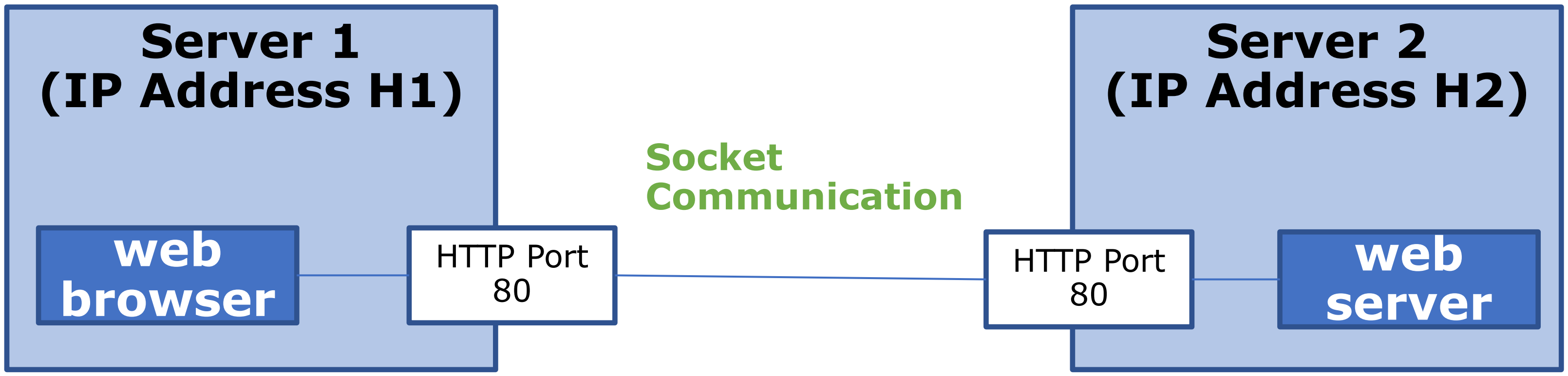
- Port
  - A software abstraction
  - 16-bit integer (0-65535)
  - Identifies a process running on a computer
  - Examples:
  - 80 (HTTP Port, allows connection to web pages from web servers)
  - 443 (HTTPS Port, allows secure connections to web pages using SSL)
  - 25 (SMTP Port, allows connection to a email server)
- Socket
  - IP Address + Port
  - Example: tux.cs.drexel.edu:80 (HTTP Port)
  - Adopted by Java, Python, etc., as the construct needed to connect to another process running on another server

Client-Server Computing



- A client process running on Server 1, with IP address H1, uses port 1 to communicate with
- A server process running on Server 2, with IP address H2, listening to port 2

Ports under 1024

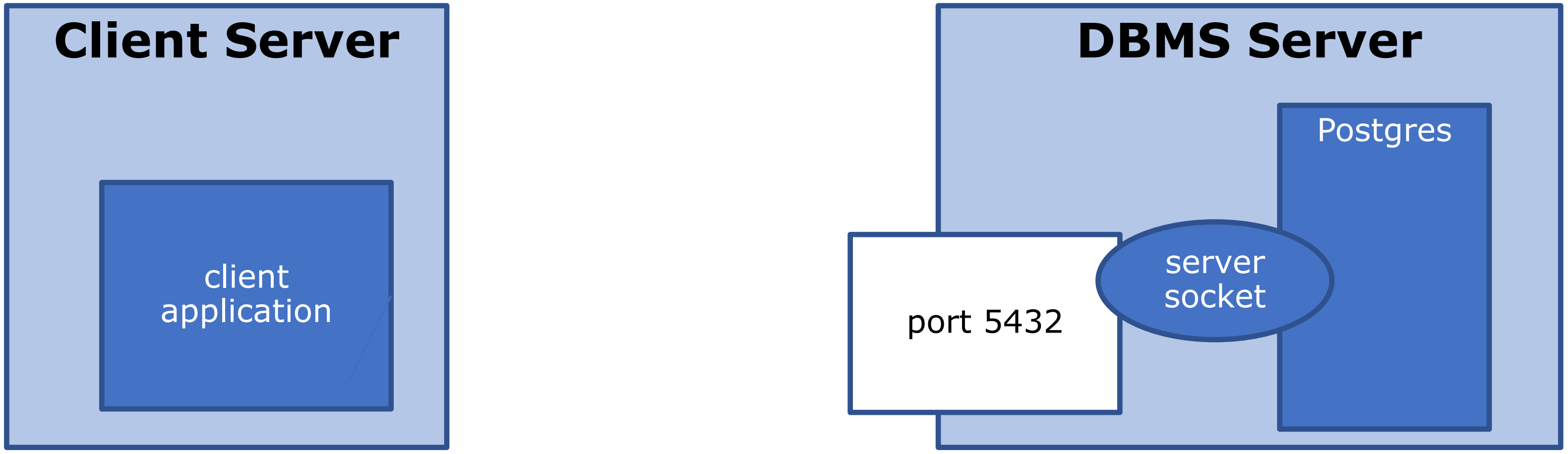


- Port numbers 0-1023 are typically reserved for special purposes, for example for protocols, such as:
- HTTP traffic is bi-directional and uses port 80
- HTTPS uses 413
- FTP uses 21
- SSH uses 22
- etc.

Database Ports

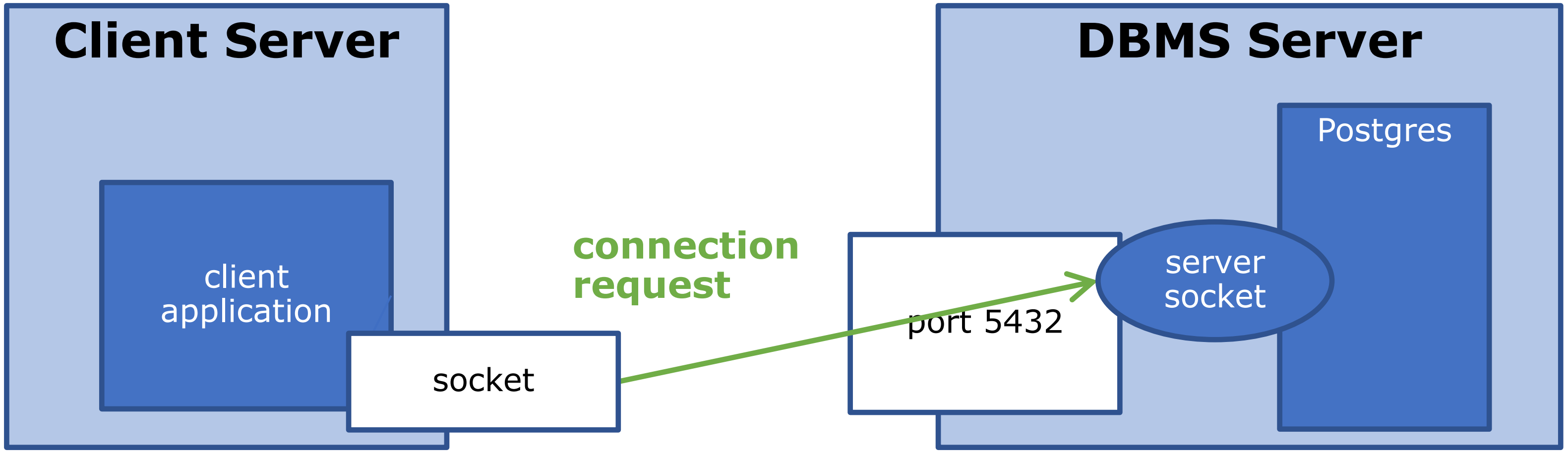
- The default port number depends on the database technology
- Postgres uses 5432
- SQL Server uses 1433
- Oracle uses 1521
- etc.

Establishing a database connection



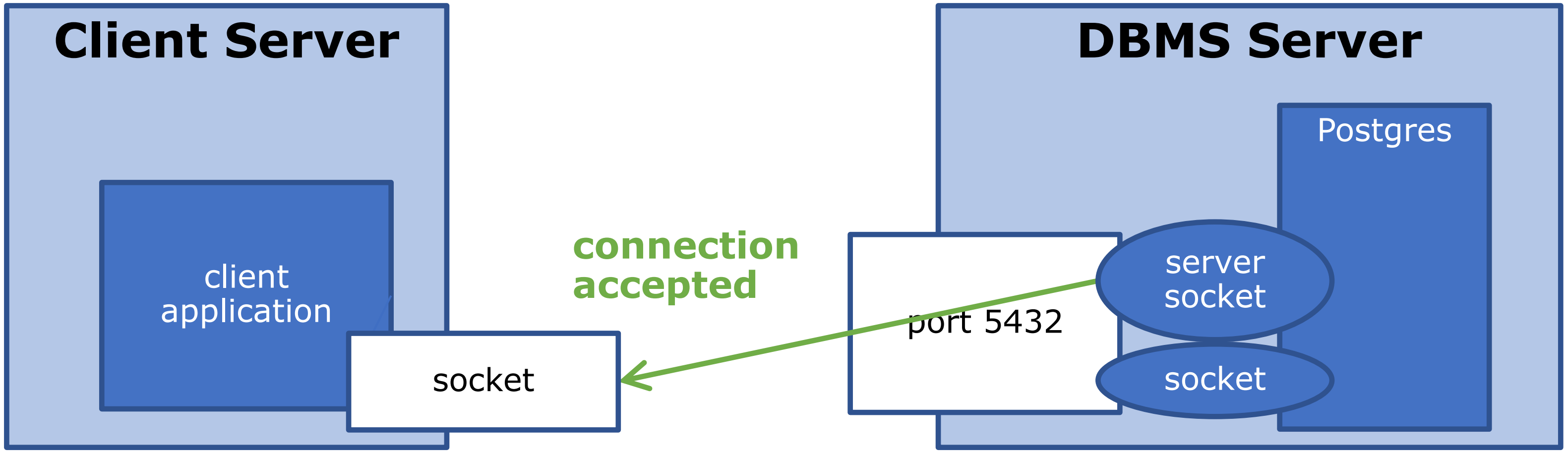
- The server process (Postgres) running on a DBMS Server creates a server socket that is listening at port 5432 for database connection requests

Establishing a database connection



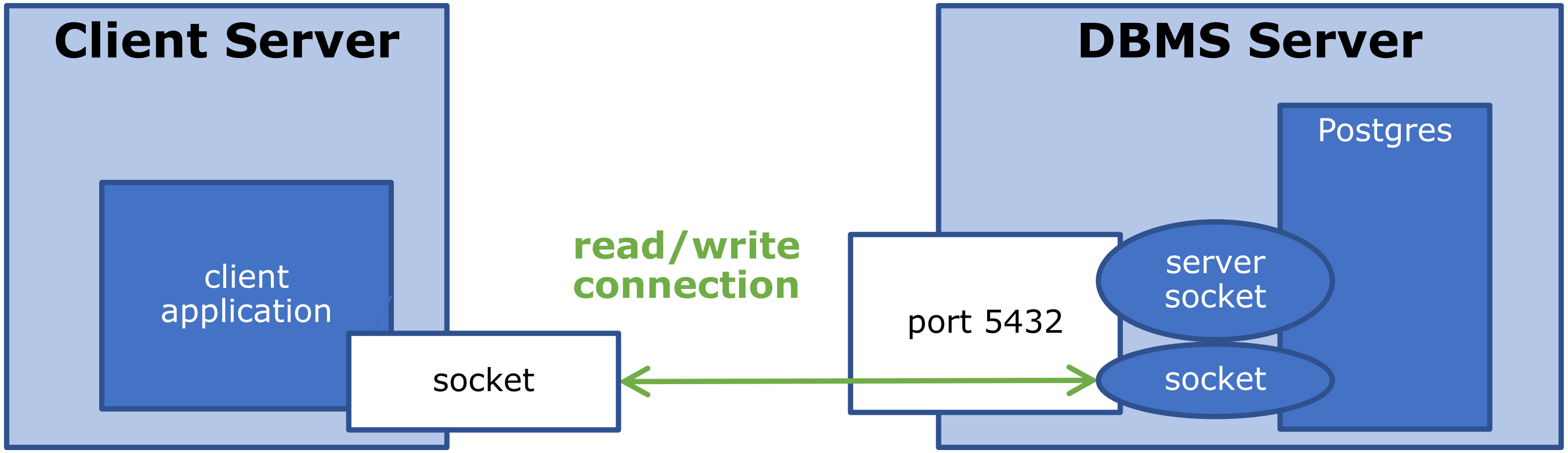
- The client process (application) creates an ephemeral socket on the client server
- The client process (application) requests a connection to the server process (Postgres) on port 5432

Establishing a database connection



- The server process (Postgres) accepts the request, creates a specialized socket for the connection and informs the client about the new socket
- In practice the server process has a pool of sockets from which it returns one (connection pool)

Establishing a database connection



- The client writes to its socket, therefore the server
- The server reads from its socket, therefore from the client
- The server writes to its socket, therefore the client
- The server reads from its socket, therefore the server

In reality…

- It is a bit more complicated
- Because of
  - Firewalls
  - Connection Pooling
  - Middleware layer
  - Etc.

So far: Database vs DBMS

- A DBMS is a system that manages a database
  - It runs on a known server
  - It listens on a known port
  - It manages one or more databases
- A database is a collection of data from a given “domain”

A Database Example – Philly Products

Customer (cid, name, city)

cid	name	city
1	Kevin Bacon	Philadelphia
2	Kevin Hart	Philadelphia
3	Grace Kelly	Philadelphia

Product (pid, name, price)

pid	name	price
1	Pretzel	1.25
2	TastyKake	2.75
3	Peanut Chews	5.99
4	Campbell Soup	1.78
5	Breyers Ice Cream	4.42

Orders (cid, pid, qty)

cid	pid	qty
1	1	12
1	5	1
2	3	2
2	4	2
2	5	2

SQL

- Structured Query Language
- ISO Standard
- De facto language for communicating with a relational database

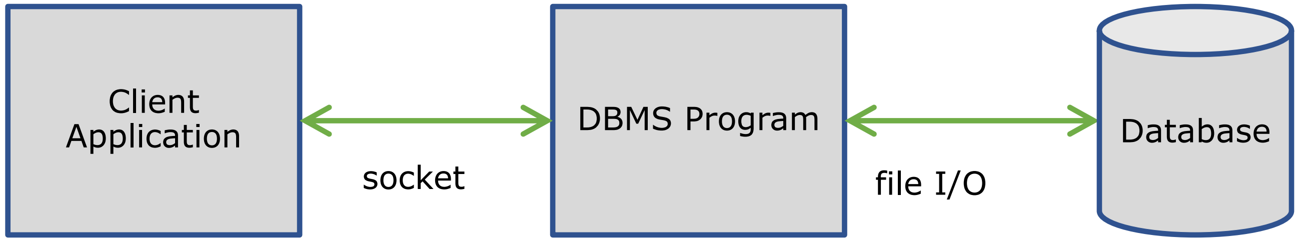
DB Technology

- Relational Databases
  - Oracle
  - mySQL
  - IBM DB2
  - Microsoft SQL Server
  - Hive
  - SQLite
  - ..

There are 339 different DB technologies https://db-engines.com/en/ranking

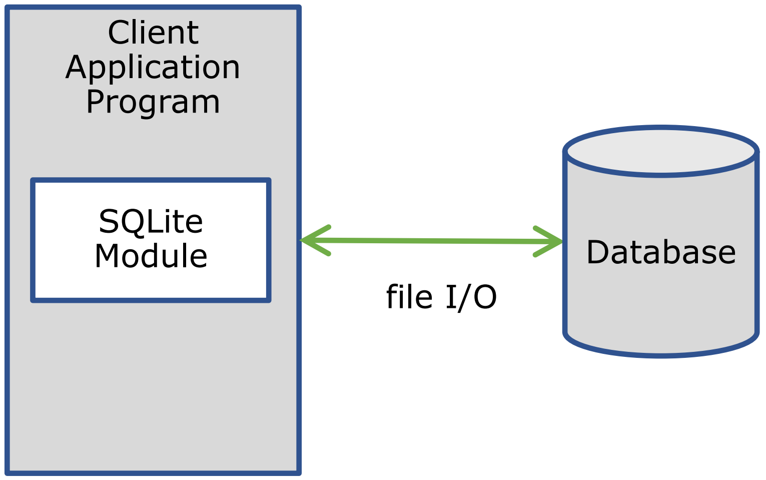
- NoSQL Databases
  - Key-Value Stores: Redis, DynamoDB, ..
  - Document Stores: MongoDB, DynamoDB, ..
  - Graph DBMS: Neo4j, Amazon Neptune, …
- SQLite
  - Popular and free DBMS Software
  - Lightweight
  - Uses / extends SQL

Typical Relational DBMS (RDBMS)



- The client application program and the DBMS program are typically running on different machines
- They communicate using sockets

Typical SQLite



- The DBMS is a module
- The client application and the DBMS module run in the same process
- They communicate using function calls

Using SQLite Command Line Client

- Use sqlite3 with a filename to create and/or access a SQLite database
- It is case-sensitive
- SQLite statements begin with a "."

```
.help
.quit
.tables
.schema
.schema [table]
```

```
% sqlite3 philly.sqlite
SQLite version 3.13.0 2016-05-18 10:57:30
Enter ".help" for usage hints.
sqlite> .quit
```

SQLite datatypes

SQLite	Java	Python
INTEGER	short, int or long	int
REAL	double or float	float
TEXT	String	str
BLOB	Object	object
NULL type	null type	NoneType

CREATE TABLE

CREATE TABLE [IF NOT EXISTS] table (column datatype, …);

```
sqlite> create table Customer (cid INTEGER, name TEXT, city TEXT);
sqlite> create table Product(pid INTEGER, name TEXT, price REAL);
sqlite> create table Orders(cid INTEGER, pid INTEGER, qty INTEGER);
sqlite> create table Map(a INTEGER, b integer);
sqlite> .tables
Customer  Map        Orders     Product
```

DROP TABLE

DROP TABLE [IF EXISTS] table;

```
sqlite> drop table map;
sqlite> .tables
Customer  Orders     Product
```

ALTER TABLE

ALTER TABLE table specification [,specification]…;

ALTER TABLE table ADD column column-type;

ALTER TABLE table ADD column column-type AFTER another;

```
sqlite> alter table Orders add Date TEXT;
sqlite> .schema Orders
CREATE TABLE Orders(cid INTEGER, pid INTEGER, qty INTEGER, Date
TEXT);
```

CREATE INDEX

CREATE [UNIQUE] INDEX indexname ON table (column);

```
sqlite> create unique index idx_cid on Orders(cid);
sqlite> .indexes
idx_cid
sqlite> .schema Orders
CREATE TABLE Orders(cid INTEGER, pid INTEGER, qty INTEGER, Date
TEXT);
CREATE UNIQUE INDEX idx_cid on Orders(cid);
sqlite> create unique index idx_cust_cid on Customer(cid);
sqlite> .schema Customer
CREATE TABLE Customer (cid INTEGER, name TEXT, city TEXT);
CREATE UNIQUE INDEX idx_cust_cid on Customer(cid);
sqlite> .indexes
idx_cid
idx_cust_cid
```

Add Data

INSERT INTO table (column, …) VALUES (expr, …);

```
sqlite> select * from Customer;
sqlite> insert into Customer(cid,name,city) values (1,'Kevin
Bacon','Philadelphia');
sqlite> insert into Customer values (2,'Kevin
Hart','Philadelphia');
sqlite> insert into Customer values (3,'Grace
Kelly','Philadelphia');
sqlite> select * from Customer;
1|Kevin Bacon|Philadelphia
2|Kevin Hart|Philadelphia
3|Grace Kelly|Philadelphia
sqlite> insert into Customer values (1,'Dimitra Vista',
'Philadelphia');
Error: UNIQUE constraint failed: Customer.cid
```

Add Data (continued)

INSERT INTO table (column, …) VALUES (expr, …);

```
sqlite> insert into Product values (1,'Pretzel',1.25);
sqlite> insert into Product values (2,'TastyKake',2.75);
sqlite> insert into Product values (3,'Peanut Chews',5.99);
sqlite> insert into Product values (4,'Campbell Soup',1.78);
sqlite> insert into Product values (5,'Breyers Ice Cream',4.42);
sqlite> drop table Orders;
sqlite> create table Orders(cid INTEGER, pid INTEGER, qty INTEGER);
sqlite> insert into Orders values (1,1,12);
sqlite> insert into Orders values (1,5,1);
sqlite> insert into Orders values (2,3,2);
sqlite> insert into Orders values (2,4,2);
sqlite> insert into Orders values (2,5,2);
sqlite> select * from Orders;
1|1|12
1|5|1
2|3|2
2|4|2
2|5|2
```

Delete Data

DELETE FROM table WHERE condition;

```
sqlite> delete from Customer where cid=3;
sqlite> select * from Customer;
1|Kevin Bacon|Philadelphia
2|Kevin Hart|Philadelphia
```

Update Data

UPDATE table SET column1=expr1 [, column2=expr2 …]

[WHERE condition];

```
sqlite> update Customer set name='Dimitra Vista' where cid=1;
sqlite> select * from Customer;
1|Dimitra Vista|Philadelphia
2|Kevin Hart|Philadelphia
```

Select Data

SELECT [DISTINCT] expr, … FROM table, …

[WHERE condition]

[ORDER BY column [ASC | DESC]];

```
sqlite> select * from Customer;
1|Dimitra Vista|Philadelphia
2|Kevin Hart|Philadelphia
sqlite> select * from Customer order by name DESC;
2|Kevin Hart|Philadelphia
1|Dimitra Vista|Philadelphia
sqlite> select * from Customer where name like 'Kevin%';
2|Kevin Hart|Philadelphia
sqlite> select * from Customer where name and cid = 2; like
'Kevin%'
2|Kevin Hart|Philadelphia
```

Select Data - Continued

SELECT [DISTINCT] expr, … FROM table, …

[WHERE condition]

[ORDER BY column [ASC | DESC]];

```
sqlite> .schema Orders
CREATE TABLE Orders(cid INTEGER, pid INTEGER, qty INTEGER);
sqlite> select * from Orders;
1|1|12
1|5|1
2|3|2
2|4|2
2|5|2
sqlite> select max(qty) from Orders;
12
sqlite> select distinct(cid) from Orders;
1
2
```

Join Table

SELECT column1, column2, …

FROM table 1, table2, …

WHERE condition

```
sqlite> select * from Customer, Product, Orders where
Orders.cid=Customer.cid and Orders.pid=Product.pid;
1|Dimitra Vista|Philadelphia|1|Pretzel|1.25|1|1|12
1|Dimitra Vista|Philadelphia|5|Breyers Ice Cream|4.42|1|5|1
2|Kevin Hart|Philadelphia|3|Peanut Chews|5.99|2|3|2
2|Kevin Hart|Philadelphia|4|Campbell Soup|1.78|2|4|2
2|Kevin Hart|Philadelphia|5|Breyers Ice Cream|4.42|2|5|2
```

Read from a file

.read filename

(filename must contain sqlite statements)

```
sqlite> .read Customer.sql


sqlite> select * from Customer;
1|Kevin Bacon|Philadelphia
2|Kevin Hart|Philadelphia
3|Grace Kelly|Philadelphia
```

Import from a csv file

.import filename table

(filename must be a csv file)

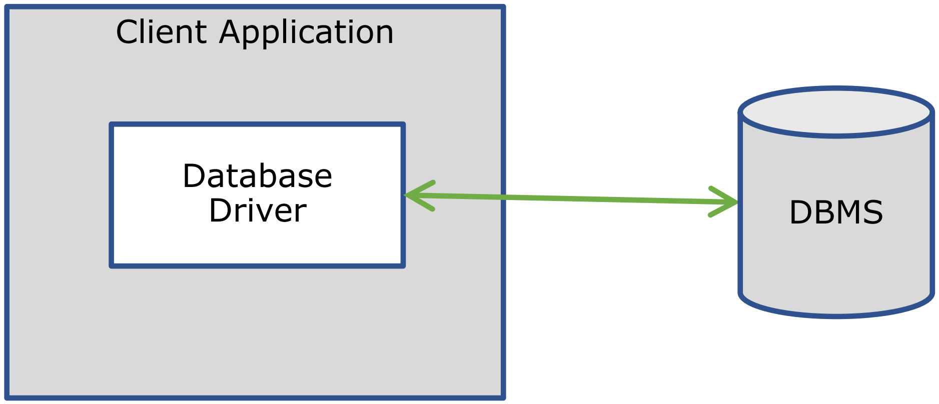
```
sqlite> .mode csv
sqlite> .import nba.csv nba
```

Using SQLite programmatically

- From Java
- From Python

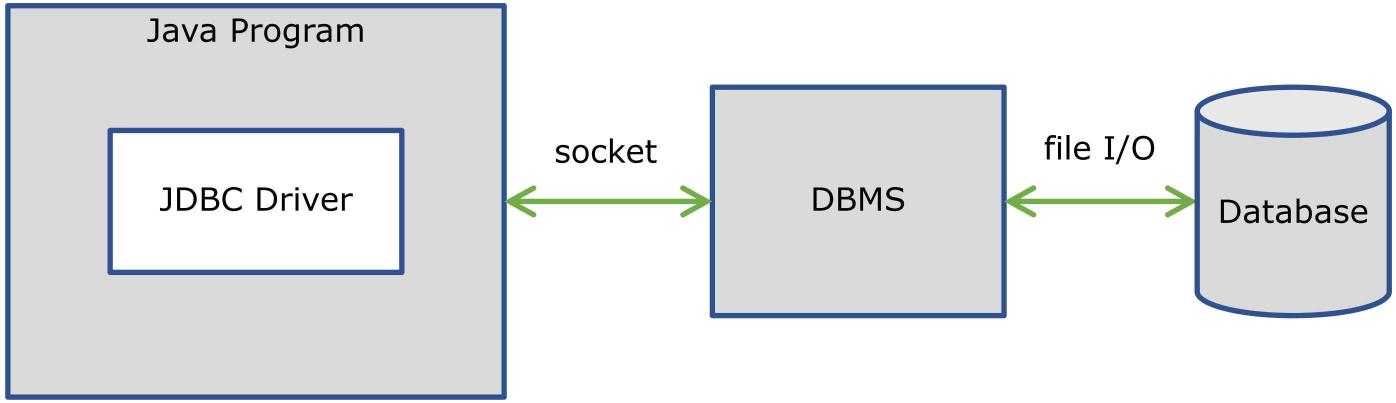
Database Drivers

- A client application program must use a Database Driver to connect to a DBMS



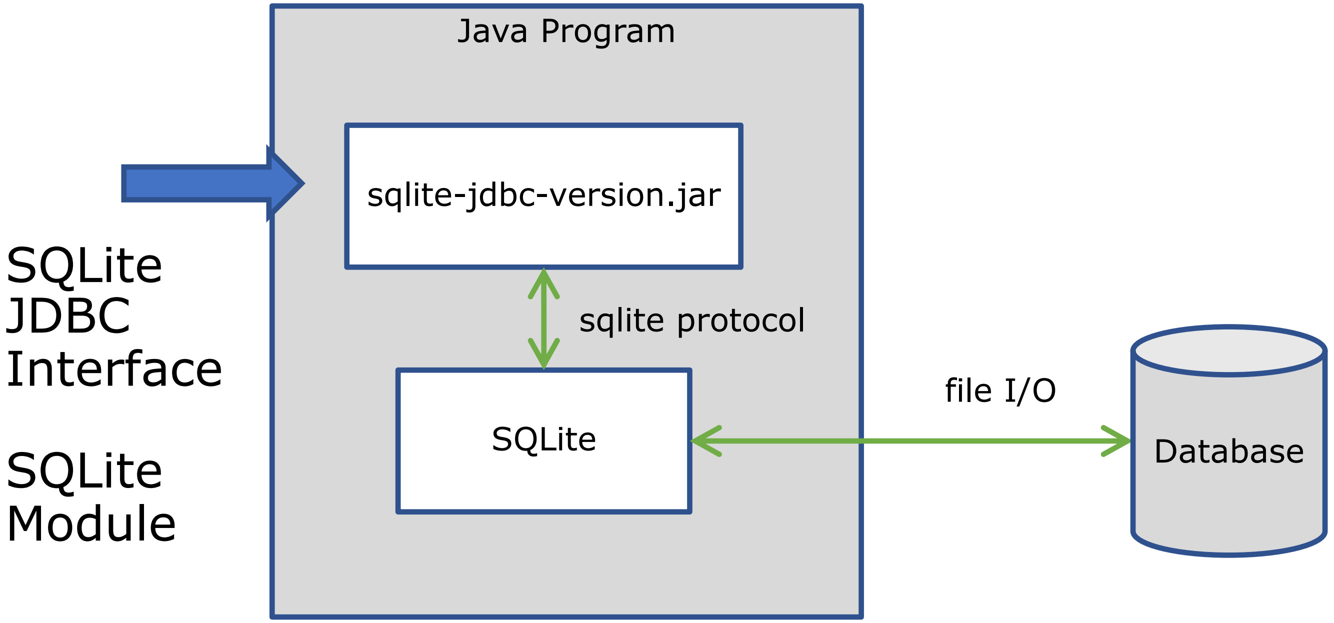
Database Drivers for Java

- Java uses the Java Database Connectivity (JDBC) Interface
- JDBC Drivers are vendor specific & DBMS-specific
- For example, there is an Oracle JDBC Driver



Using SQLite from Java

- sqlite-jdbc-version.jar must be in CLASSPATH



Java Database Connection URLs

To establish a connection to a database from Java using the JDBC interface you need a database URL, which varies depending on the DBMS

- MySQL
  - `jdbc:mysql://localhost:3306/philly`
  - where `jdbc` is the API, `mysql` is the DBMS technology, `localhost` is the server where MySQL is running, `3306` is the port number and `philly` is the name of the database
- Oracle
  - `jdbc:oracle:thin:@localhost:1521:philly`
  - where `jdbc` is the API, `oracle` is the DBMS, `thin` is the driver, `localhost` is the server name on which oracle is running, we may also use IP address, `1521` is the port number and `philly` is the database
- *SQLite*
  - *`jdbc:sqlite:C:/sqlite/db/philly.db`*
  - where `jdbc` is the API, `sqlite` is the DBMS technology, and `C:/sqlite/db/philly.db` is the path to the database

Java code

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


public class Demo
{
    public static void main(String[] args) {
        Connection conn = null;
        Statement stmt;
        ResultSet rs;
        String query;


        try {
            conn = DriverManager.getConnection("jdbc:sqlite:philly.db");
            System.out.println("Connection established");


            stmt = conn.createStatement();
            query = "SELECT CID, NAME, CITY FROM Customer";
```

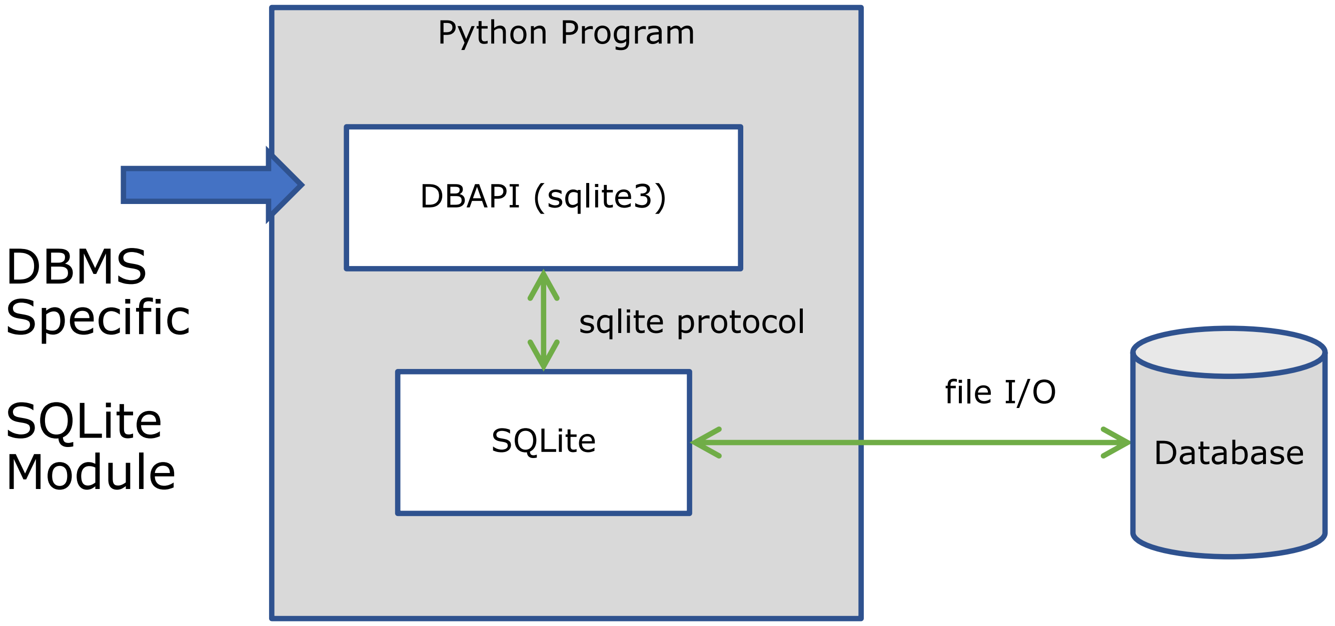
Java code – cont’d

```
            rs = stmt.executeQuery(query);
            System.out.println("Queries executed");


            while (rs.next()) {
                System.out.print(rs.getInt("CID")
                                + ":" + rs.getString("NAME")
                                + ":" + rs.getString("CITY")
                                + "\n");
            }
        }
        catch(SQLException e) {
            System.err.println(e.getMessage());
        }
        finally {
            try {
                if(conn != null)
                    conn .close();
            } catch(SQLException e) {
                System.err.println(e.getMessage());
            }
       }
    }
}
```

Using SQLite from Python

- DBAPI is a DBMS-specific implementation
- `sqlite3` module must be installed
- Typically bundled with python though



Python Code

```
import sqlite3
conn = sqlite3.connect('philly.sqlite')

c = conn.cursor()

c.execute("DELETE FROM Customer where cid='5'")
c.execute("INSERT INTO Customer VALUES ('5','Humpty
Dumpty','Philadelphia')")


for row in c.execute('SELECT * FROM Customer order by name'):
    print(row)

conn.commit()
conn.close()
```

Resources (FOR REFERENCE ONLY)

- Resources this lecture
  - Class Notes
- JDBC & SQLite
  - JDBC Oracle Documentation https://www.oracle.com/technetwork/java/javase/tech/index-jsp-136101.html
  - JDBC Tutorials https://docs.oracle.com/javase/tutorial/jdbc/index.html
  - SQLite Usage https://github.com/xerial/sqlite-jdbc#usage
- Database Engines
  - https://db-engines.com/en/ranking