JDBC

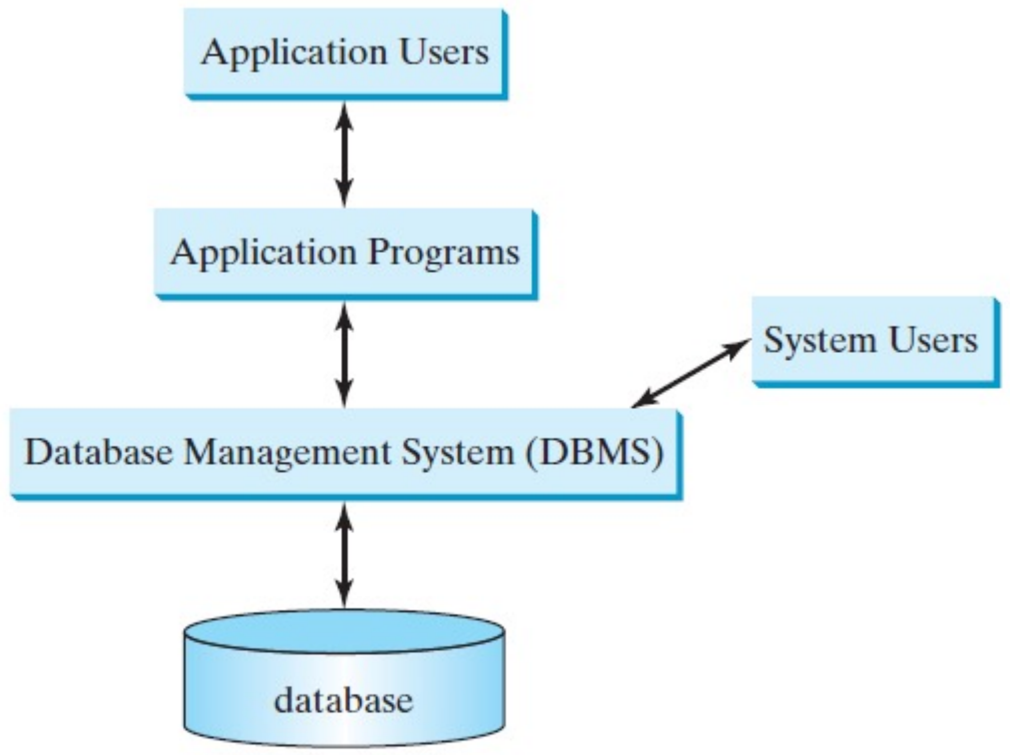
Dr.Praisan padungweang

School of information and technology

KMUTT

Database

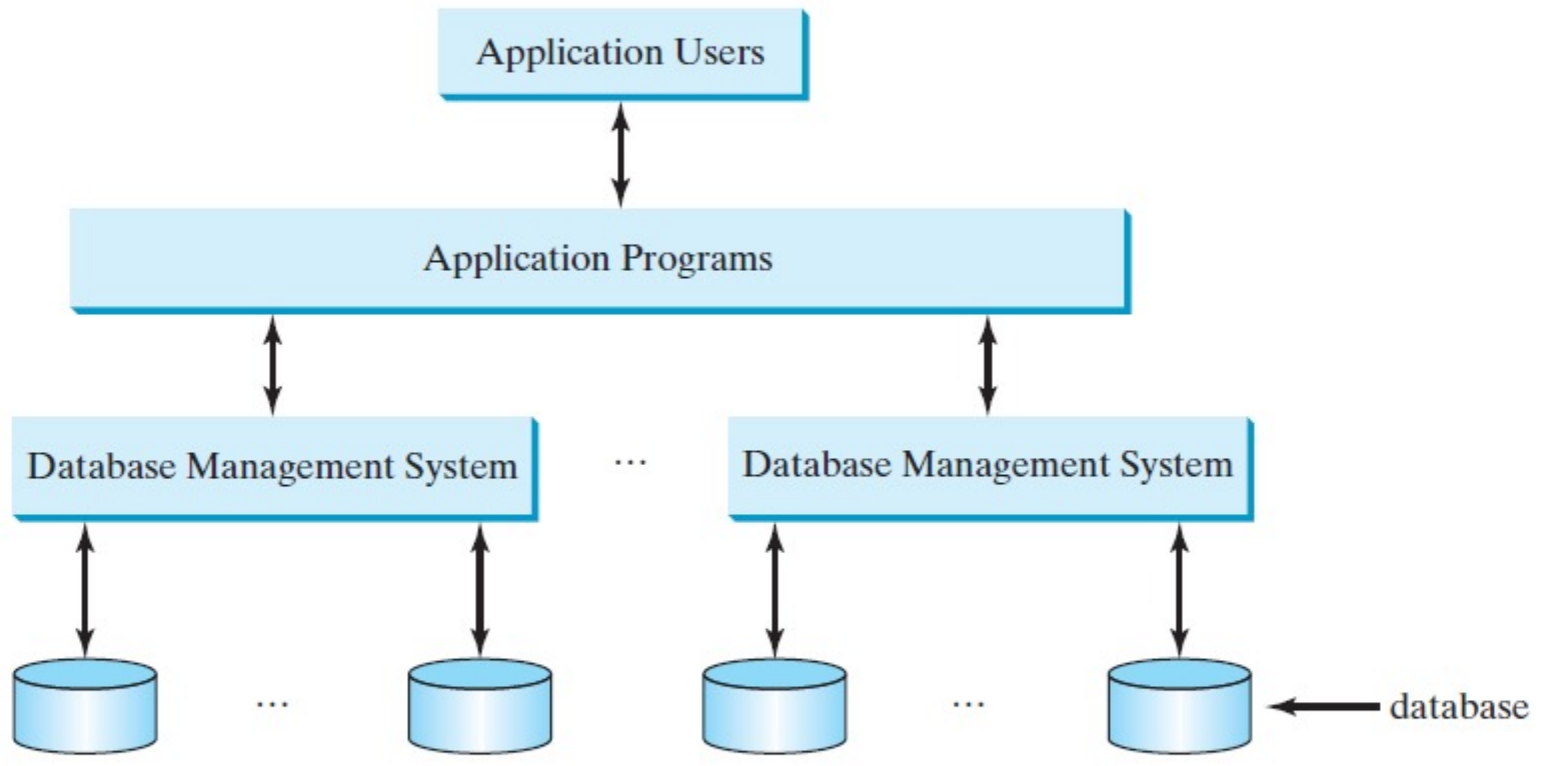
A database system consists of data, database management software, and application programs.



Y. Daniel Liang, "Java Programming Comprehensive Version Tenth Edition"

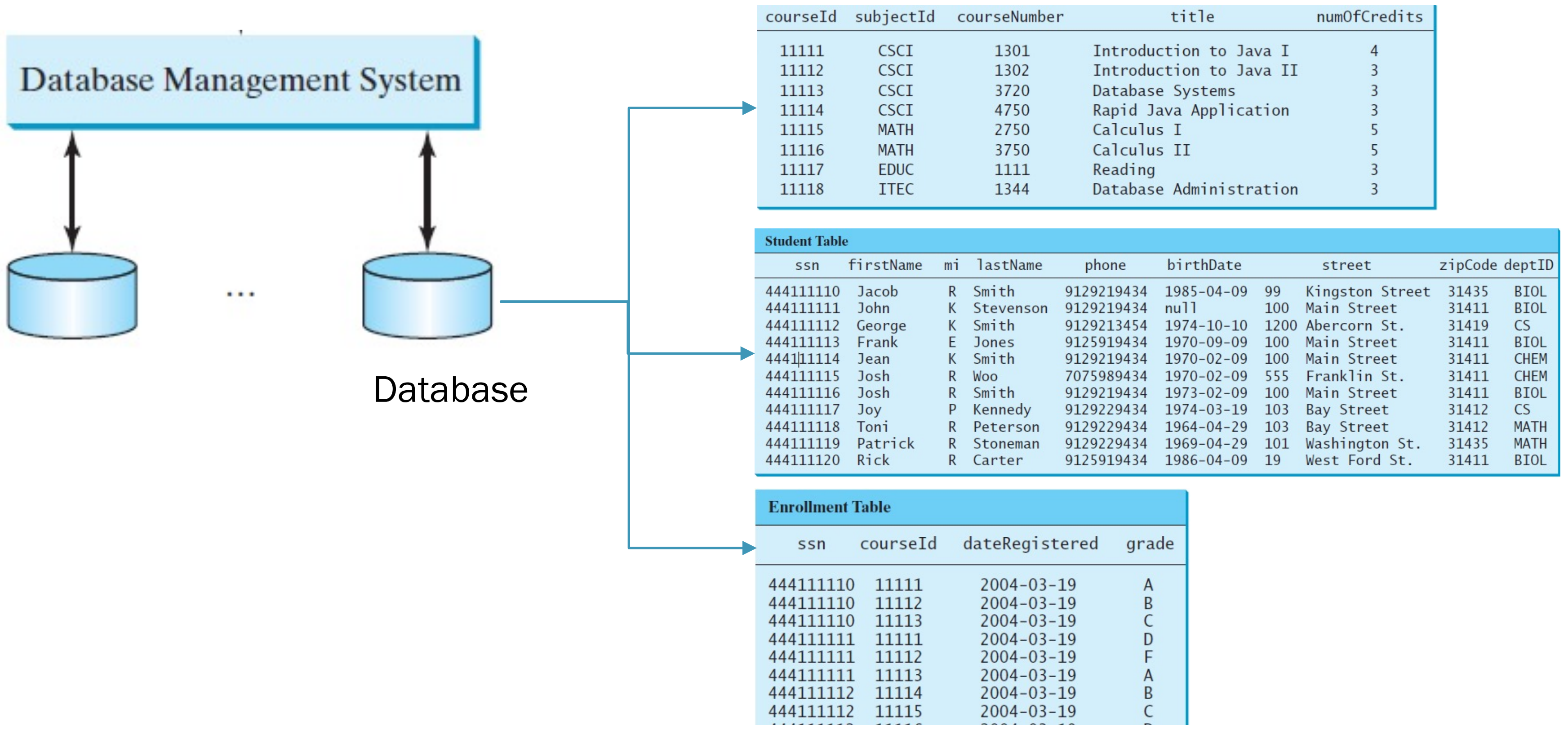
Application software

An application program can access multiple database systems.



Structured Query Language (SQL) is a standard computer language for relational database management and data manipulation.

Table in RDBMS



courseId	subjectId	courseNumber	title	numOfCredits
11111	CSCI	1301	Introduction to Java I	4
11112	CSCI	1302	Introduction to Java II	3
11113	CSCI	3720	Database Systems	3
11114	CSCI	4750	Rapid Java Application	3
11115	MATH	2750	Calculus I	5
11116	MATH	3750	Calculus II	5
11117	EDUC	1111	Reading	3
11118	ITEC	1344	Database Administration	3

Student Table

ssn	firstName	mi	lastName	phone	birthDate	street	zipCode	deptID
444111110	Jacob	R	Smith	9129219434	1985-04-09	99 Kingston Street	31435	BIOL
444111111	John	K	Stevenson	9129219434	null	100 Main Street	31411	BIOL
444111112	George	K	Smith	9129213454	1974-10-10	1200 Abercorn St.	31419	CS
444111113	Frank	E	Jones	9125919434	1970-09-09	100 Main Street	31411	BIOL
444111114	Jean	K	Smith	9129219434	1970-02-09	100 Main Street	31411	CHEM
444111115	Josh	R	Woo	7075989434	1970-02-09	555 Franklin St.	31411	CHEM
444111116	Josh	R	Smith	9129219434	1973-02-09	100 Main Street	31411	BIOL
444111117	Joy	P	Kennedy	9129229434	1974-03-19	103 Bay Street	31412	CS
444111118	Toni	R	Peterson	9129229434	1964-04-29	103 Bay Street	31412	MATH
444111119	Patrick	R	Stoneman	9129229434	1969-04-29	101 Washington St.	31435	MATH
444111120	Rick	R	Carter	9125919434	1986-04-09	19 West Ford St.	31411	BIOL

Enrollment Table

ssn	courseId	dateRegistered	grade
444111110	11111	2004-03-19	A
444111110	11112	2004-03-19	B
444111110	11113	2004-03-19	C
444111111	11111	2004-03-19	D
444111111	11112	2004-03-19	F
444111111	11113	2004-03-19	A
444111112	11114	2004-03-19	B
444111112	11115	2004-03-19	C

Tables

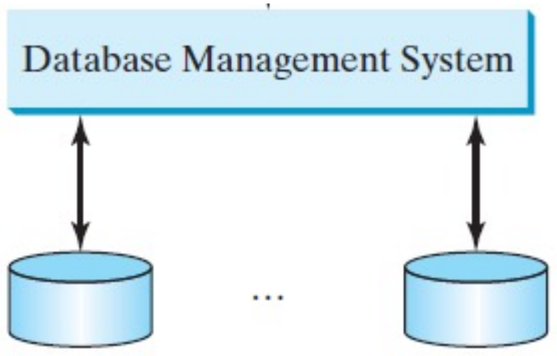
Table in RDBMS

Columns/Attributes

Tuples/Rows/Records

ID	NAME
620001	Chris Touchton
620002	Donna Close
620003	Deadra Nims
620004	Deidra Landin
620005	Patrick Fraise

SQL



student_club

```
CREATE DATABASE student_club;

CREATE TABLE student (
   id INT not null primary key,
   name VARCHAR(50)
);

INSERT INTO student VALUES(620001, 'Chris Touchton');

SELECT * FROM student;
```

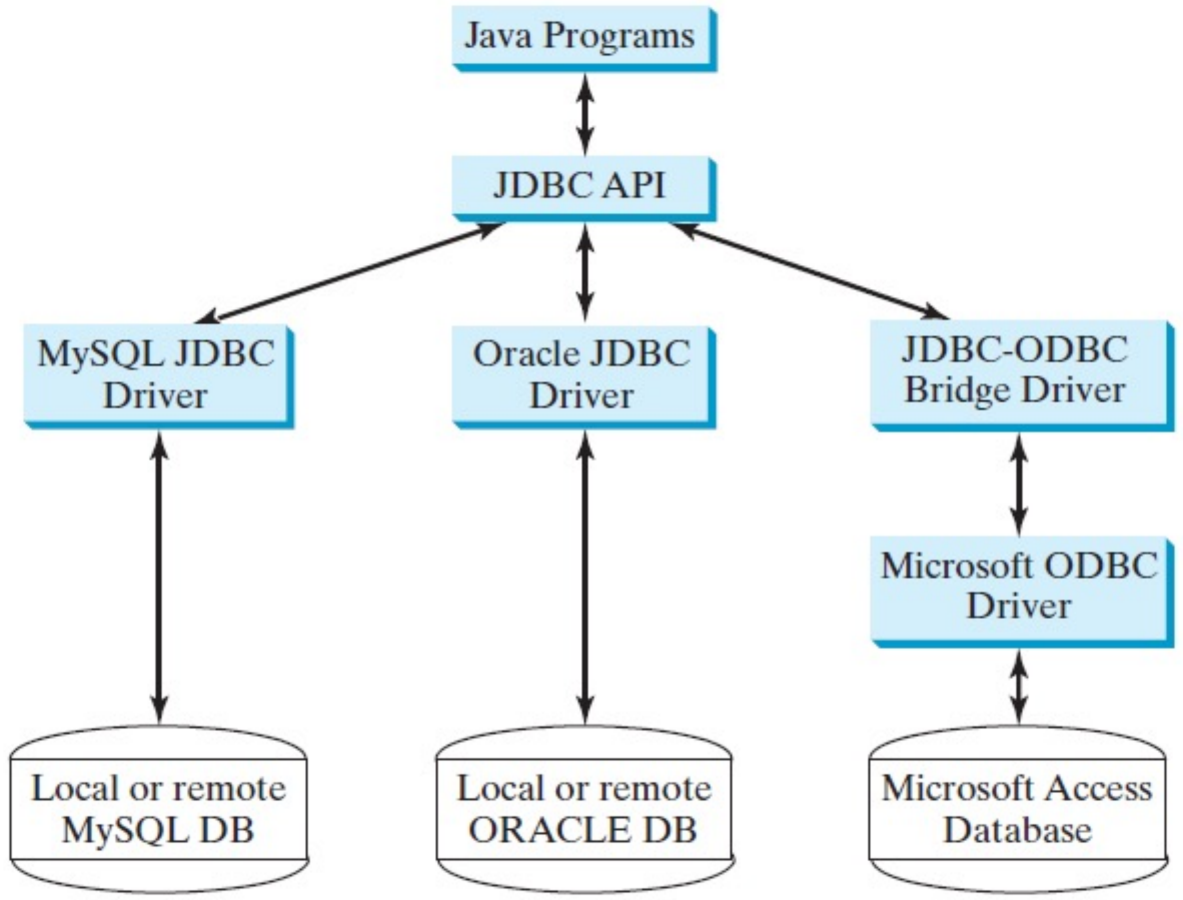
student

primary key

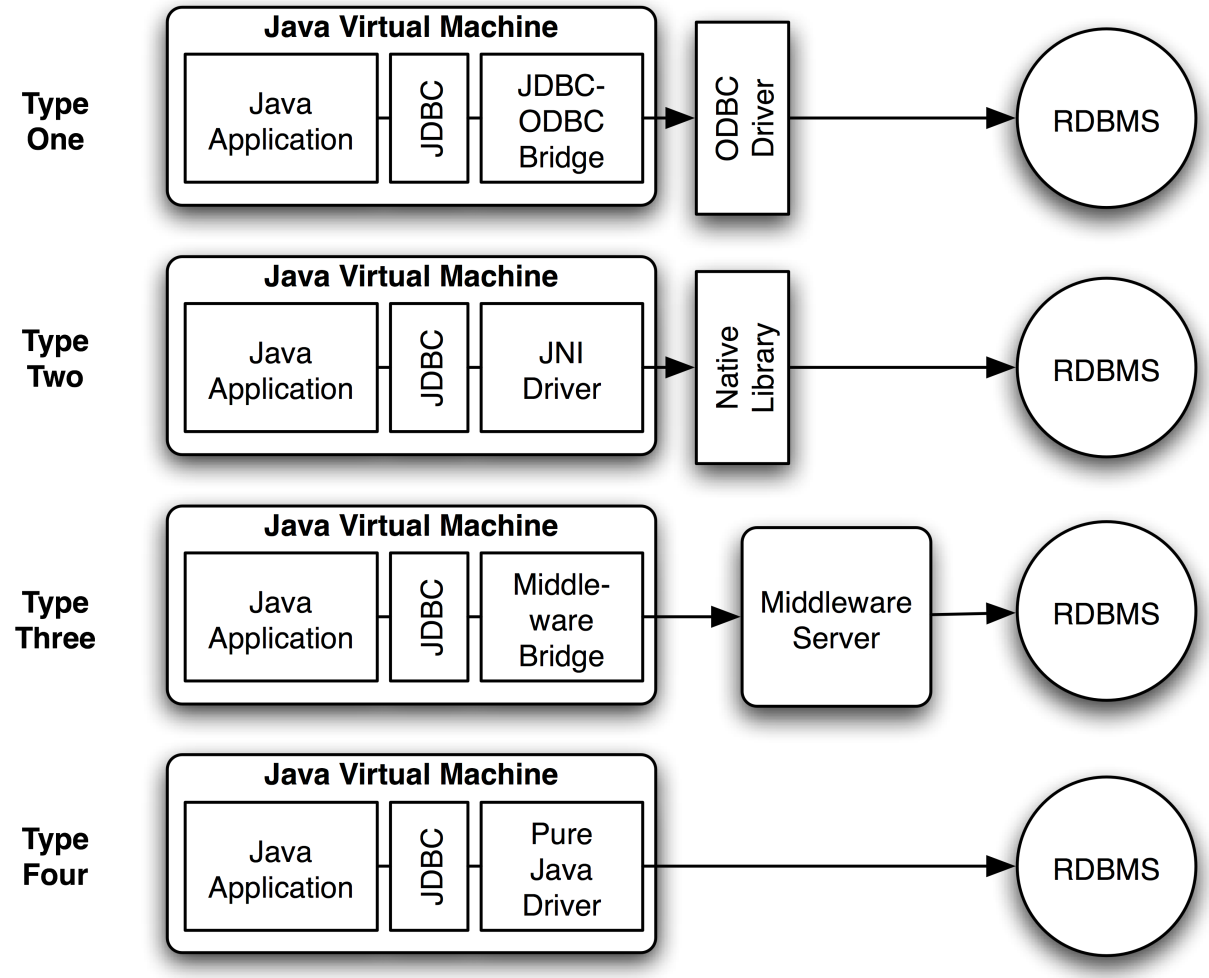
ID	NAME
620001	Chris Touchton

Java Database Connectivity (JDBC)

Java programs access and manipulate databases through JDBC drivers.



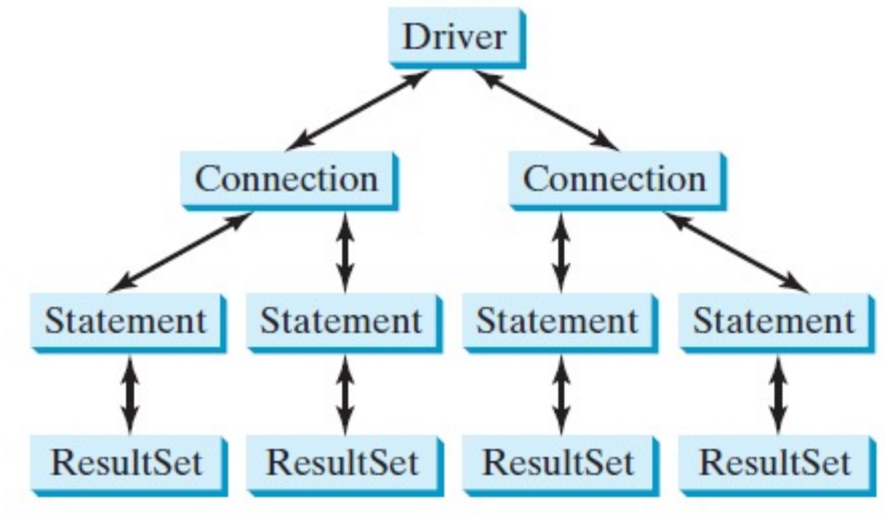
JDBC Drivers



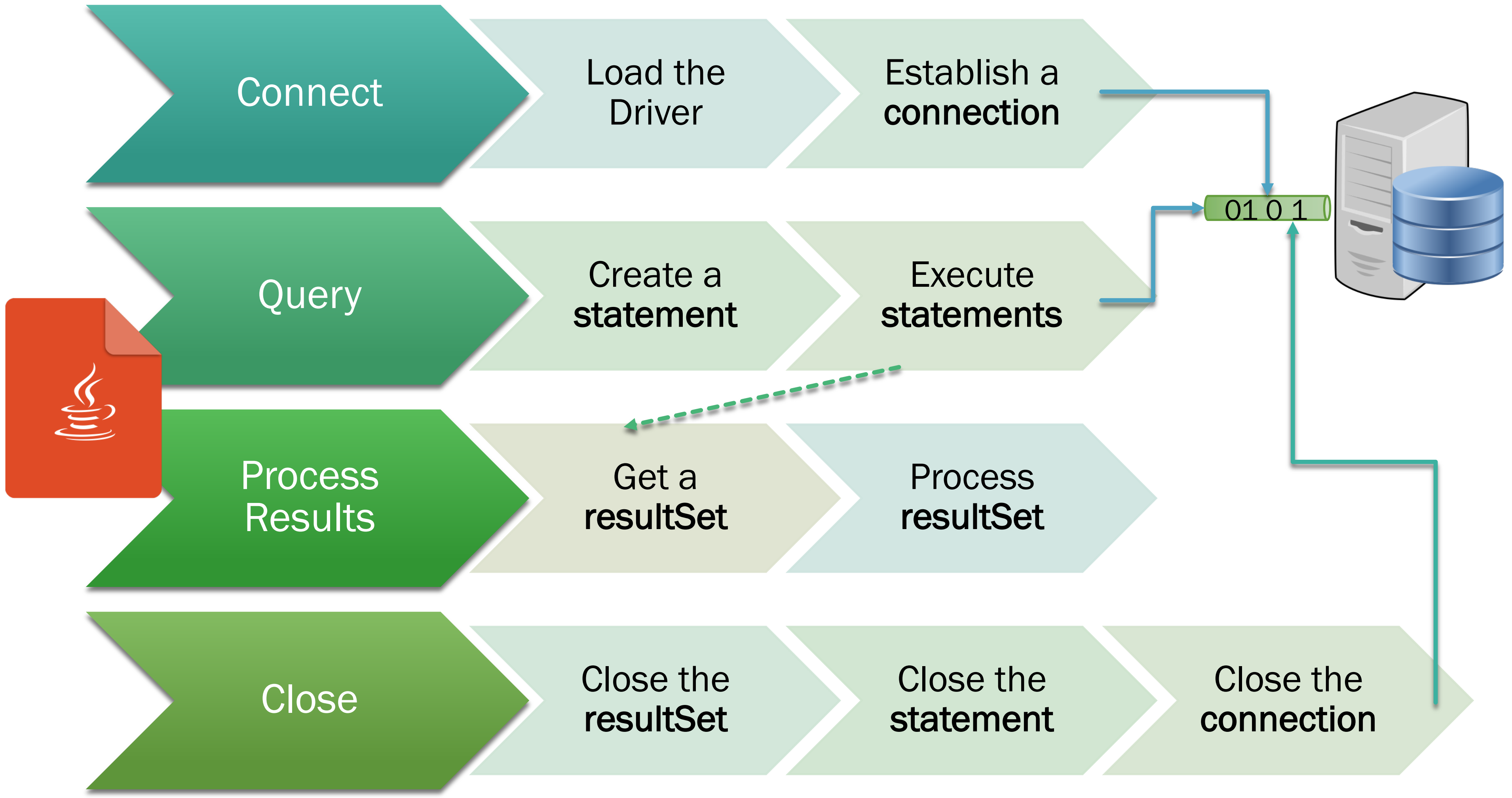
https://github.com/ikite/ikite.github.io/wiki/Java-Database-Connectivity-(JDBC)

Java Database Connectivity (JDBC)

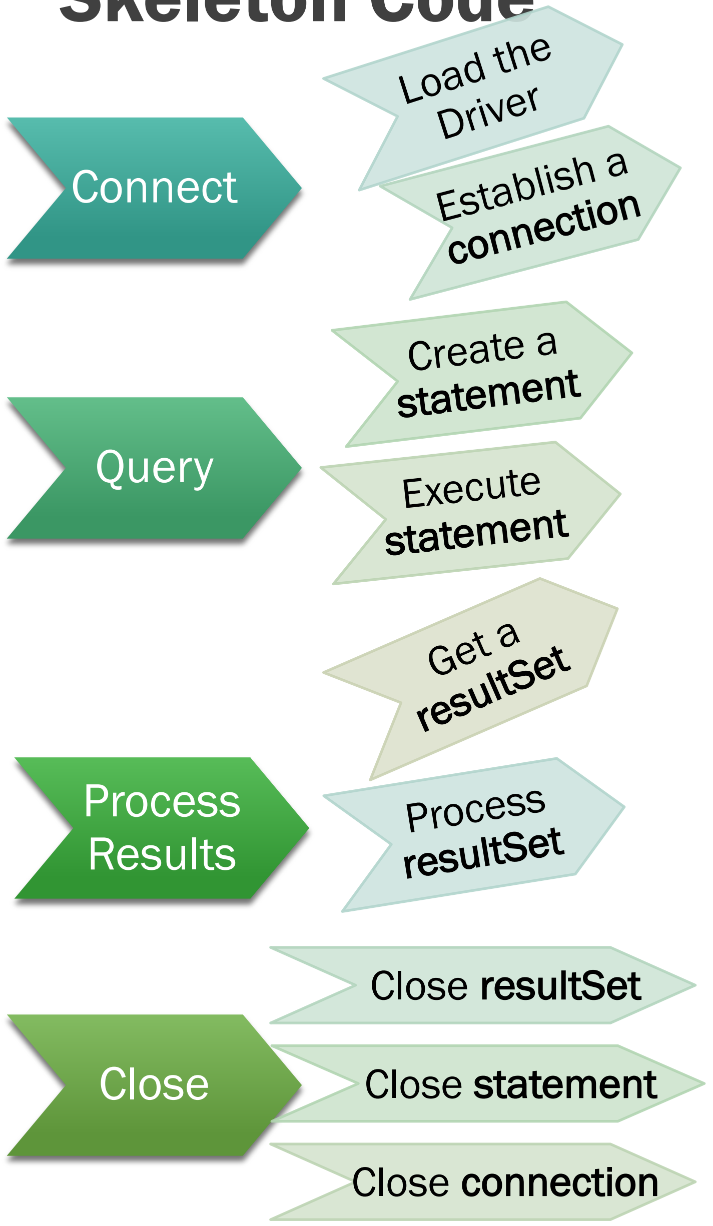
JDBC classes enable Java programs to connect to the database, send SQL statements, and process results.



Basic Steps to Use a Database in Java

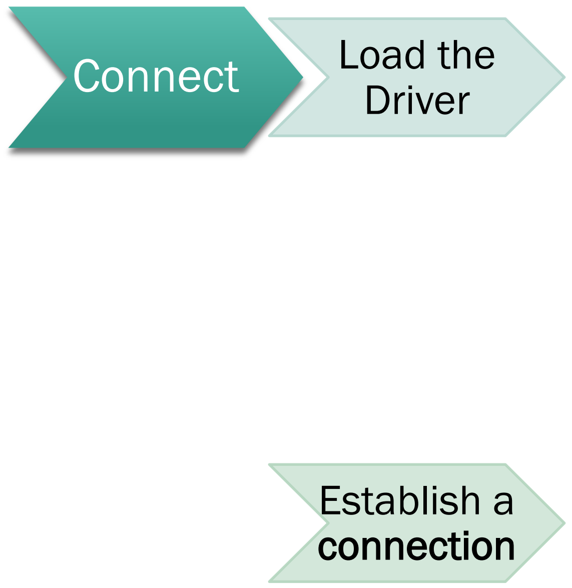


Skeleton Code



```
Class.forName(DRIVERNAME);

Connection con = DriverManager.getConnection(

                    CONNECTIONURL,
                    DBID, DBPASSWORD);

Statement stmt = con.createStatement();

ResultSet rs = stmt.executeQuery( “SELECT a, b, c FROM member”);

while(rs.next()) {

      int x = rs.getInt(“a”);

      String s = rs.getString(“b”);

      float f = rs.getFloat(“c”);

}

rs.close();   stmt.close();   con.close();
```

Basic Steps to Use a Database in Java



```
Class.forName(DRIVERNAME);
```

Database	Driver Class	Source
derby	org.apache.derby.jdbc.ClientDriver	Already in JDK
Access	sun.jdbc.odbc.JdbcOdbcDriver	Already in JDK
MySQL	com.mysql.jdbc.Driver	mysqljdbc.jar
Oracle	oracle.jdbc.driver.OracleDriver	classes12.jar

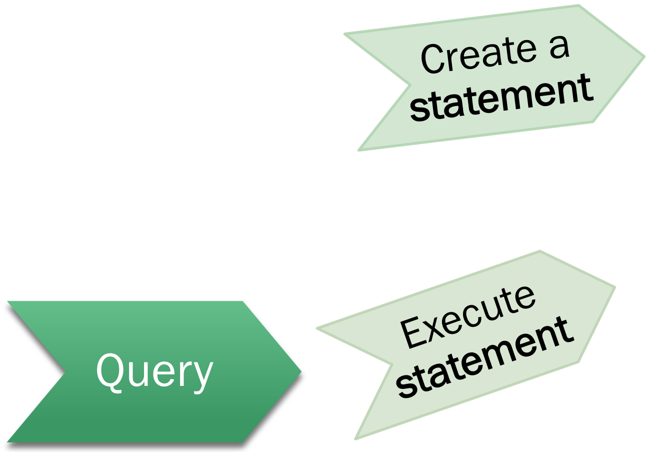
```
Connection con = DriverManager.getConnection(
                   CONNECTIONURL,
                   DBID, DBPASSWORD);
```

CONNECTIONURL	
derby	jdbc:derby://hostname:port/dbname
Access	jdbc:odbc:dataSource
MySQL	jdbc:mysql://hostname/dbname
Oracle	jdbc:oracle:thin:@hostname:port#:oracleDBSID

The Derby database

- Download Derby: https://mirror.csclub.uwaterloo.ca/apache/db/derby/db-derby-10.14.2.0/
  Choose the lib.zip file.
- Unzip this to any location you choose.
- In NetBeans,
  - select the Services tab,
  - right-click on JavaDB,
  - select Properties
    Java DB installation -> the unzipped Derby library folder.
    Databases location -> you can choose any folder that you want to store your database files

Basic Steps to Use a Database in Java



```
Statement stmt = con.createStatement();

stmt.executeUpdate("CREATE TABLE customer
            (id INT NOT NULL,
            name VARCHAR(100),
            PRIMARY KEY(id))");

int ins_rows = stmt.executeUpdate(
            "INSERT INTO customer
             VALUES (001,'Georgia Lucas')");

int del_rows = stmt.executeUpdate(
            "DELETE FROM customer
             WARE id=001");

ResultSet rs = stmt.executeQuery(
             "SELECT id, name FROM customer");
```

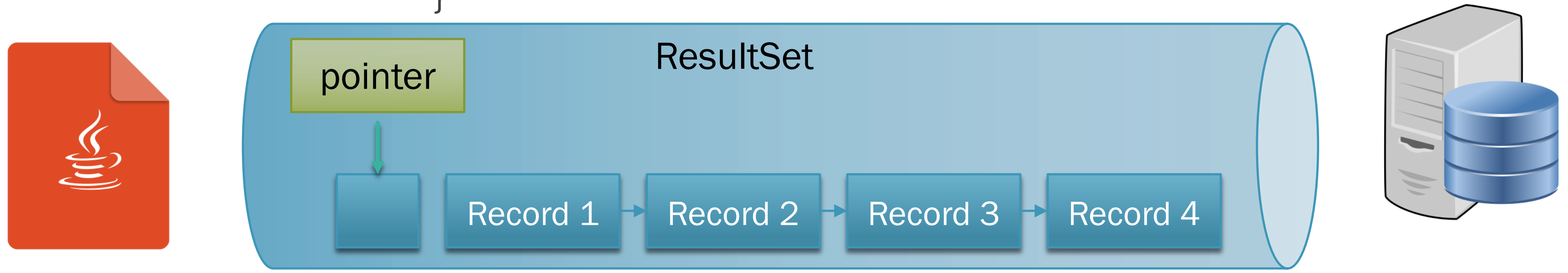
The execute, executeQuery, and executeUpdate Methods

- The executeQuery method should be used if the execution produces a single result set, such as the SQL SELECT statement.
- The executeUpdate method should be used if the statement results in a single update count or no update count, such as a SQL INSERT, DELETE, UPDATE, or DDL (Data definition language) statement.

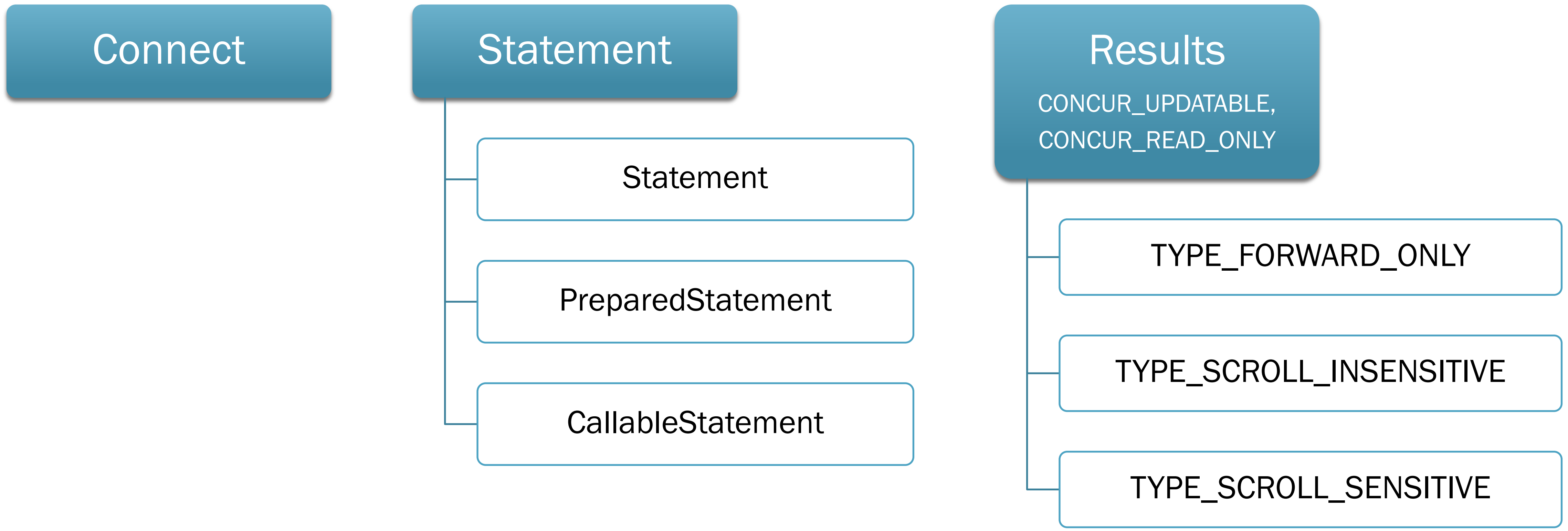
Basic Steps to Use a Database in Java

Get a resultSet
Process Results
Process resultSet

```
ResultSet rs = stmt.executeQuery(
          “SELECT id, name FROM customer”);


List<Customer> cust_list=new ArrayList<Customer>();

while(rs.next()) {

    int id = rs.getInt(“id”);

    String name = rs.getString(“name”);

    cust_list.add(new Customer(id,name))

}
```



JDBC API


Connect
Statement
Statement
PreparedStatement
CallableStatement
Results
CONCUR_UPDATABLE,
CONCUR_READ_ONLY
TYPE_FORWARD_ONLY
TYPE_SCROLL_INSENSITIVE
TYPE_SCROLL_SENSITIVE

PROCESSING SQL STATEMENTS WITH JDBC

Statement Interface

- Statement interface enable us to send SQL commands and receive data from database.
- There are three different kinds of statements:
  - Statement:
    - Used to implement simple SQL statements with no parameters.
  - PreparedStatement: (Extends Statement.)
    - Used for precompiling SQL statements that might contain input parameters.
  - CallableStatement: (Extends PreparedStatement.)
    - Used to execute stored procedures that may contain both input and output parameters.
    - stored procedures -> group of SQL statements that has been created and stored in the database

Statement

- Statement accepts static SQL statements only

```
Statement stmt = con.createStatement();
int ins_rows = stmt.executeUpdate( “INSERT INTO customer VALUES (1,’Georgia Lucas’)”);
```

- PreparedStatement accepts input parameters at runtime

```
PreparedStatement pstmt = con.prepareStatement( “INSERT INTO customer VALUES(?,?)”);
pstmt.setInt(1,1);
pstmt.setString(2, “Georgia Lucas”);
pstmt.executeUpdate();
```

Executing Queries

- execute
  - Returns true if the first object that the query returns is a ResultSet object.
  - Use this method if the query could return one or more ResultSet objects.
  - Retrieve the ResultSet objects returned from the query by repeatedly calling Statement.getResultSet.
- executeQuery:
  - Returns one ResultSet object.
- executeUpdate:
  - Returns an integer representing the number of rows affected by the SQL statement.
  - Use this method if you are using INSERT, DELETE, or UPDATE SQL statements.

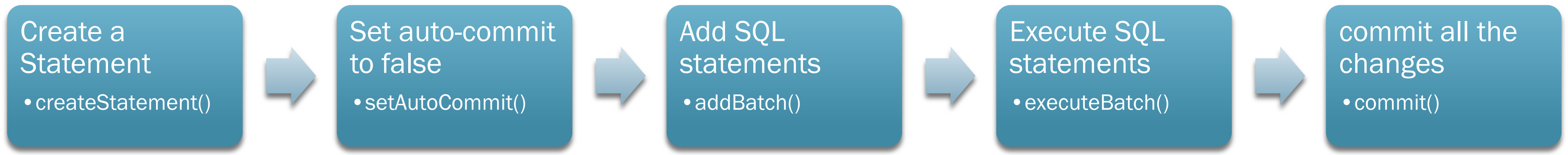
Batch Processing

Batch Processing allows to group related SQL statements into a batch and submit them with one call to the database.

Methods

- addBatch()
- executeBatch()
- clearBatch()

Steps



RESULTSET

ResultSet Interface

```
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("SELECT * FROM CUSTOMER");
```

The ResultSet interface provides methods for retrieving and manipulating the results of executed queries

This cursor is not a database cursor. This cursor is a pointer that points to one row of data in the ResultSet.

Control cursor by **next()**

Move curser to next row and returns false if the cursor is positioned after the last row

cursor

Get data at the cursor by
getInt(”cus_id”)
getString(“cus_name”)

cus_id	cus_name
7001	Georgia Lucas
7002	Dawn Banks
7003	Jeanette French
7004	Jordan Steele
7005	Bobby Fields
7006	Fernando Gross
7007	Hubert Padilla
7008	Angelina Copeland
7009	Smart watch
7010	Tina Gibson

https://www.tutorialspoint.com/jdbc/jdbc-result-sets.htm

ResultSet Types

```
Statement stmt =
            con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE);
ResultSet rs = stmt.executeQuery("SELECT * FROM CUSTOMER");
```

- TYPE_FORWARD_ONLY (default)
  - cursor moves forward only
  - The result set is insensitive to changes made to the underlying data source while it is open.
- TYPE_SCROLL_INSENSITIVE
  - cursor can move both forward and backward relative to the current position
  - cursor can move to an absolute position.
  - The result set is insensitive to changes made to the underlying data source while it is open.
- TYPE_SCROLL_SENSITIVE
  - cursor can move both forward and backward relative to the current position
  - cursor can move to an absolute position.
  - The result set reflects changes made to the underlying data source while the result set remains open.

Scrollable ResultSet

```
Statement stmt =
            con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE);
ResultSet rs = stmt.executeQuery("SELECT * FROM CUSTOMER");
```

Control pointer by
next()
previous()
absolute(int row)
relative(int row)

cursor

Get data at the pointer by
getInt(”cus_id”)
getString(“cus_name”)

cus_id	cus_name
7001	Georgia Lucas
7002	Dawn Banks
7003	Jeanette French
7004	Jordan Steele
7005	Bobby Fields
7006	Fernando Gross
7007	Hubert Padilla
7008	Angelina Copeland
7009	Smart watch
7010	Tina Gibson

https://www.tutorialspoint.com/jdbc/jdbc-result-sets.htm

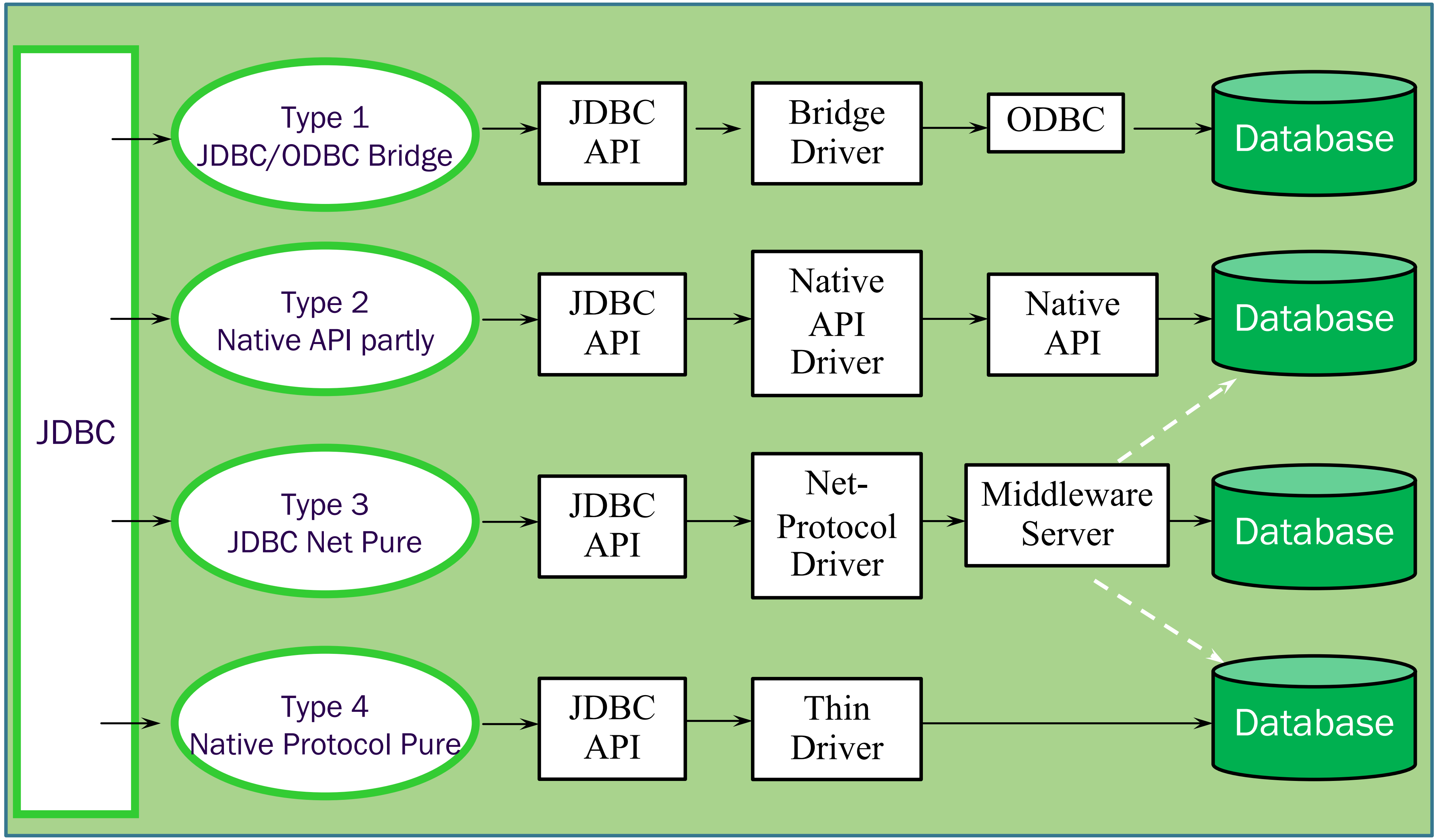
ResultSet Concurrency

- CONCUR_READ_ONLY(default)
  - The ResultSet object cannot be updated using the ResultSet interface.
- CONCUR_UPDATABLE
  - The ResultSet object can be updated using the ResultSet interface.

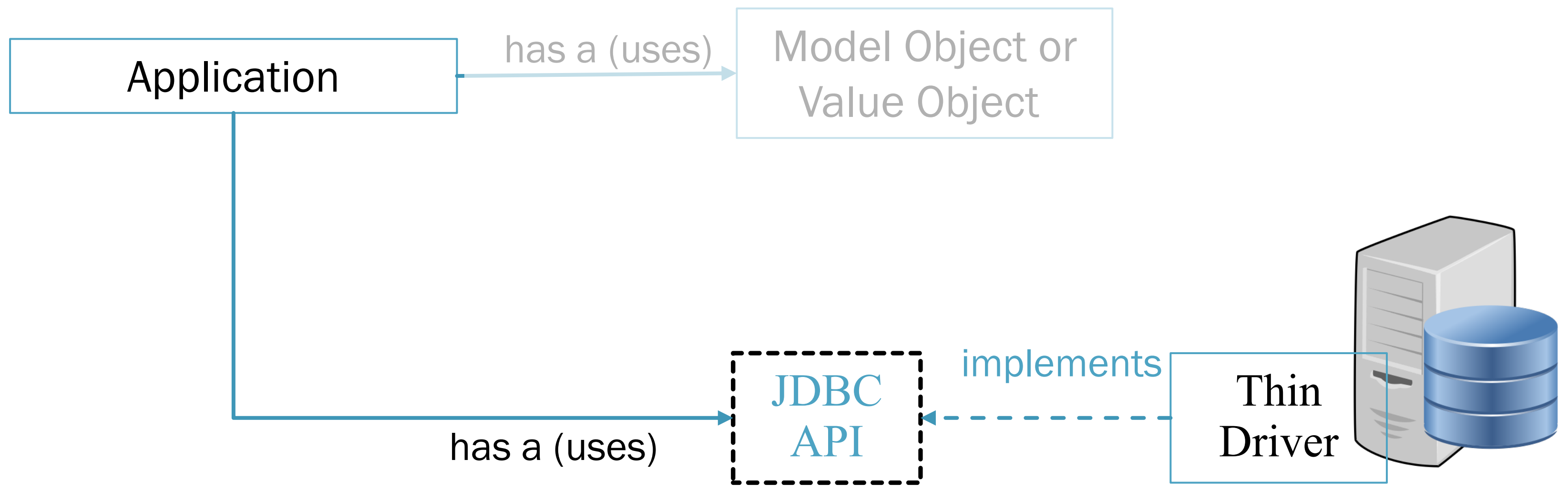
```
Statement stmt =
                con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_UPDATABLE);
 ResultSet rs = stmt.executeQuery("SELECT * FROM CUSTOMER");
```

Note:
Not all JDBC drivers and databases support concurrency.
The method DatabaseMetaData.supportsResultSetConcurrency returns true if the specified concurrency level is supported by the driver and false otherwise.

JDBC Drivers

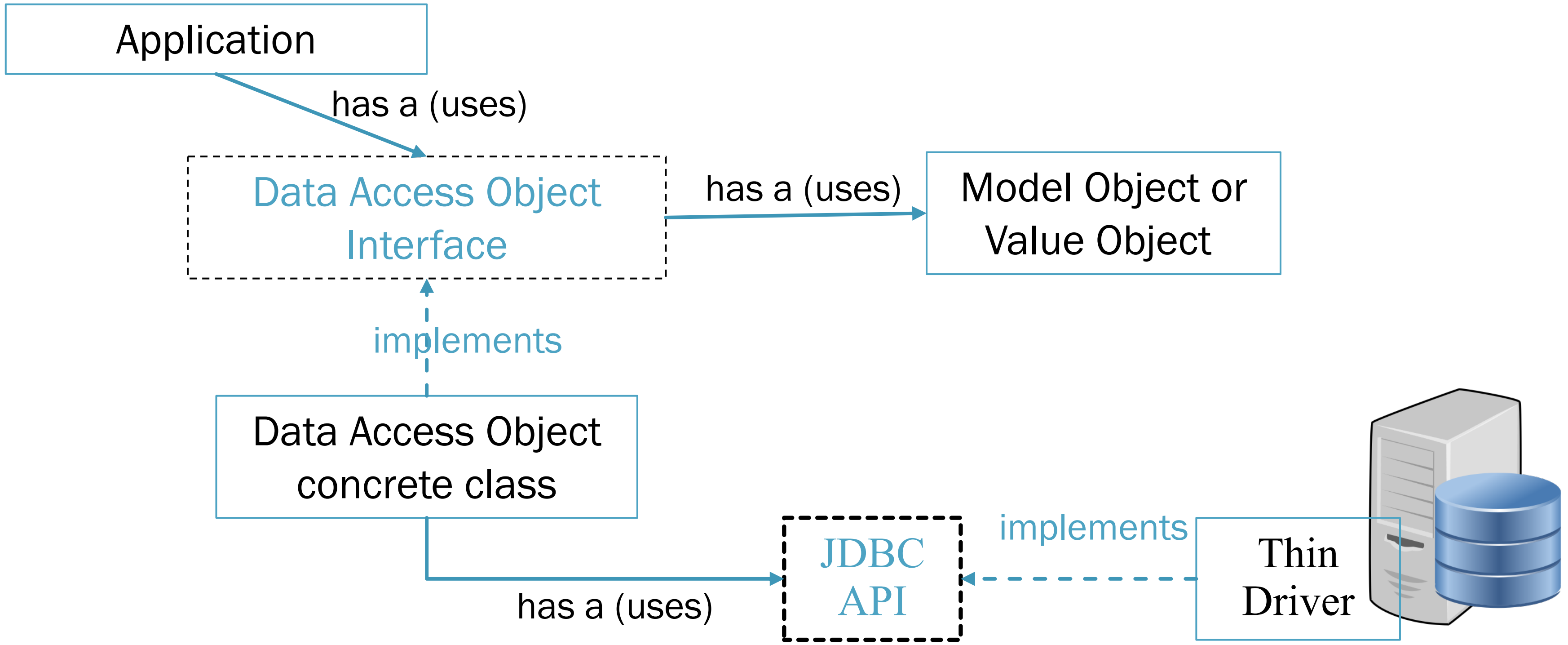


JDBC: Type 4 Native Protocol Pure



Data access object pattern (DAO)

- Separate low level data accessing API or operations from high level business services.


Application
has a (uses)
Data Access Object Interface
has a (uses)
Model Object or Value Object
implements
Data Access Object concrete class
has a (uses)
JDBC API
implements
Thin Driver