Προγραμματισμός ΙΙ (Java)

6. Διαχείριση δεδομένων

Σχεσιακές Βάσεις Δεδομένων

- Τα δεδομένα μας οργανώνονται σε ένα ή περισσότερους πίνακες: σε στήλες και σειρές
- Κάθε πίνακας έχει ένα όνομα και αποτελείται από στήλες (πεδία). Κάθε πεδίο έχει συγκεκριμένο τύπο και αποθηκεύει τιμές αυτού του τύπου
- Κάθε εγγραφή (π.χ. τα στοιχεία ενός προϊόντος) αποθηκεύεται σε μια γραμμή (πλειάδα) του πίνακα.

Film

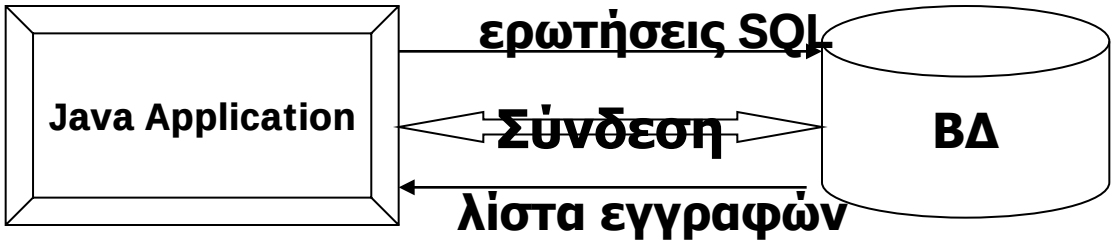
code	title	director	…	year
1150	The Godfather	Francis Ford Coppola		1972
1151	The Shawshank Redemption	Frank Darabont		1994

Συστήματα διαχείρισης ΒΔ

- Η βάση δεδομένων μπορεί να είναι ένα αρχείο ή ένα σύνολο αρχείων που το διαχειριζόμαστε μόνο με το κατάλληλο σύστημα (π.χ. MS Access, Oracle, SQL Server)
- Εκτός από το να αποθηκεύει τα δεδομένα, το ΣΔΒΔ μας επιτρέπει να υποβάλλουμε ερωτήσεις προς αυτά
  π.χ. φέρε μου τους τίτλους για τις ταινίες που βγήκαν το 1995
- Για την υποβολή ερωτήσεων χρησιμοποιούμε τη γλώσσα SQL (Structured Query Language)
- Κάθε ερώτηση σε SQL μας επιστρέφει ως απάντηση μια λίστα με πλειάδες (recordset – σύνολο εγγραφών)

Java και ΒΔ

- Για να επικοινωνήσει η εφαρμογή μας με μια βάση δεδομένων θα πρέπει:
  - Να συνδεθούμε με το ΣΔΒΔ (όπως θα ανοίγαμε ένα ρεύμα εισόδου/εξόδου προς ένα αρχείο)
  - Να συνδεθούμε με τη συγκεκριμένη ΒΔ (αν το σύστημα διατηρεί ταυτόχρονα πολλές ΒΔ).
  - Να υποβάλλουμε ερωτήσεις SQL και
  - Να διαχειριστούμε τη λίστα των εγγραφών που παίρνουμε ως απάντηση



Java Database Connectivity

- Το JDBC επιτρέπει στις εφαρμογές Java να συνδεθούν και να "ρωτήσουν" μια ΒΔ.
- Οι ερωτήσεις υποβάλλονται σε μια κοινή μορφή και μεταφράζονται στη μορφή που υποστηρίζει κάθε ΣΔΒΔ.
- Δεν υποστηρίζει ειδικές λειτουργίες του κάθε ΣΔΒΔ
- Το πακέτο java.sql περιέχει τις τάξεις για το JDBC 1.0
- Το πακέτο *javax.sql* υποστηρίζει τα JDBC 2.0 και 3.0

Java Database Connectivity API (1)

Υπάρχουν 4 κατηγορίες JDBC οδηγών:

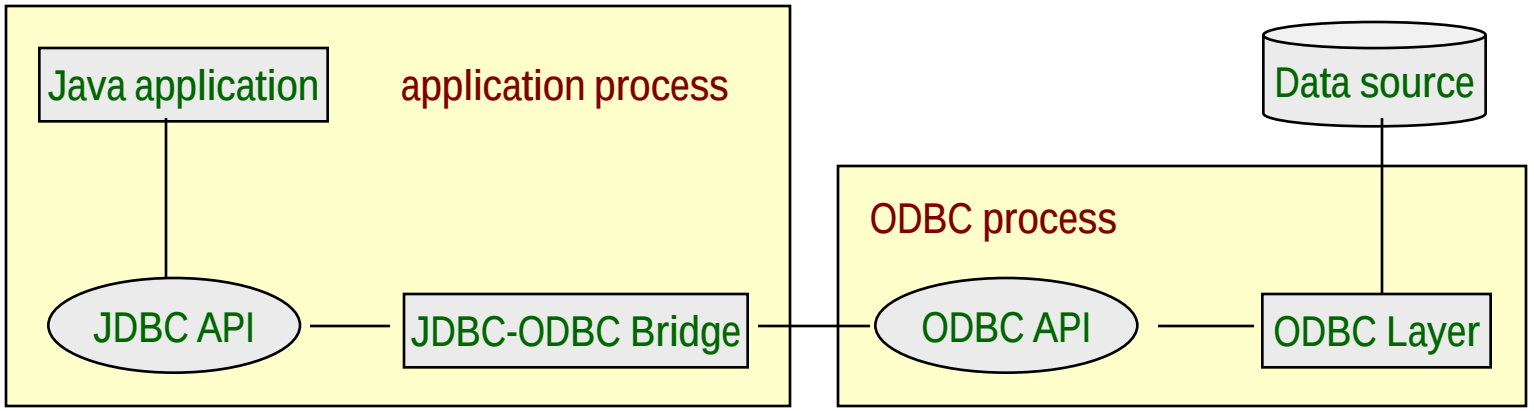
- JDBC-ODBC bridge
- Native κώδικας και JDBC
- Γενικός JDBC driver γραμμένος σε Java
- JDBC driver σε Java

Στον πρώτο τρόπο ο οδηγός έρχεται με το SDK.

Στους άλλους τρεις ο οδηγός διατίθεται από το δημιουργό του ΣΔΒΔ ή κάποιον τρίτο.

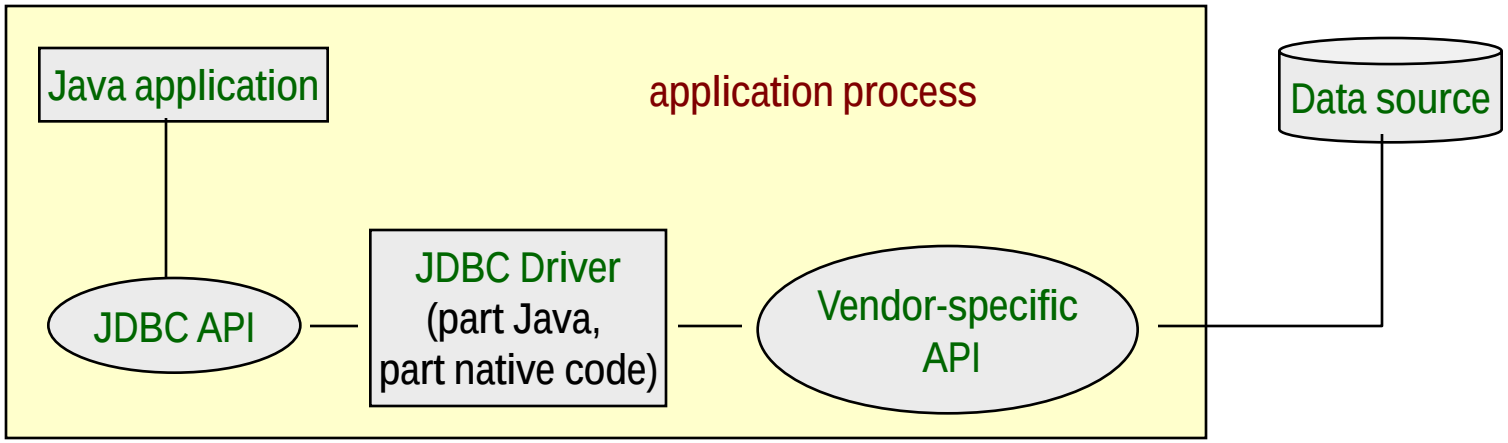
JDBC-ODBC Bridge

- Το **O**pen **D**ata**B**ase **C**onnectivity (ODBC) είναι ένα πρότυπο που ανέπτυξε η Microsoft για τα Windows.
  - Κάθε ΣΔΒΔ που εγκαθιστούμε στα windows εγκαθιστά τον αντίστοιχο ODBC οδηγό. Οι ερωτήσεις προς τον οδηγό μετατρέπονται σε κλήσεις στο ΣΔΒΔ.
- Ο JDBC-ODBC οδηγός είναι ο μόνος τρόπος για να συνδεθούμε με μια ΒΔ σε Access.
  - Ο ευκολότερος αλλά και πιο αργός τρόπος.



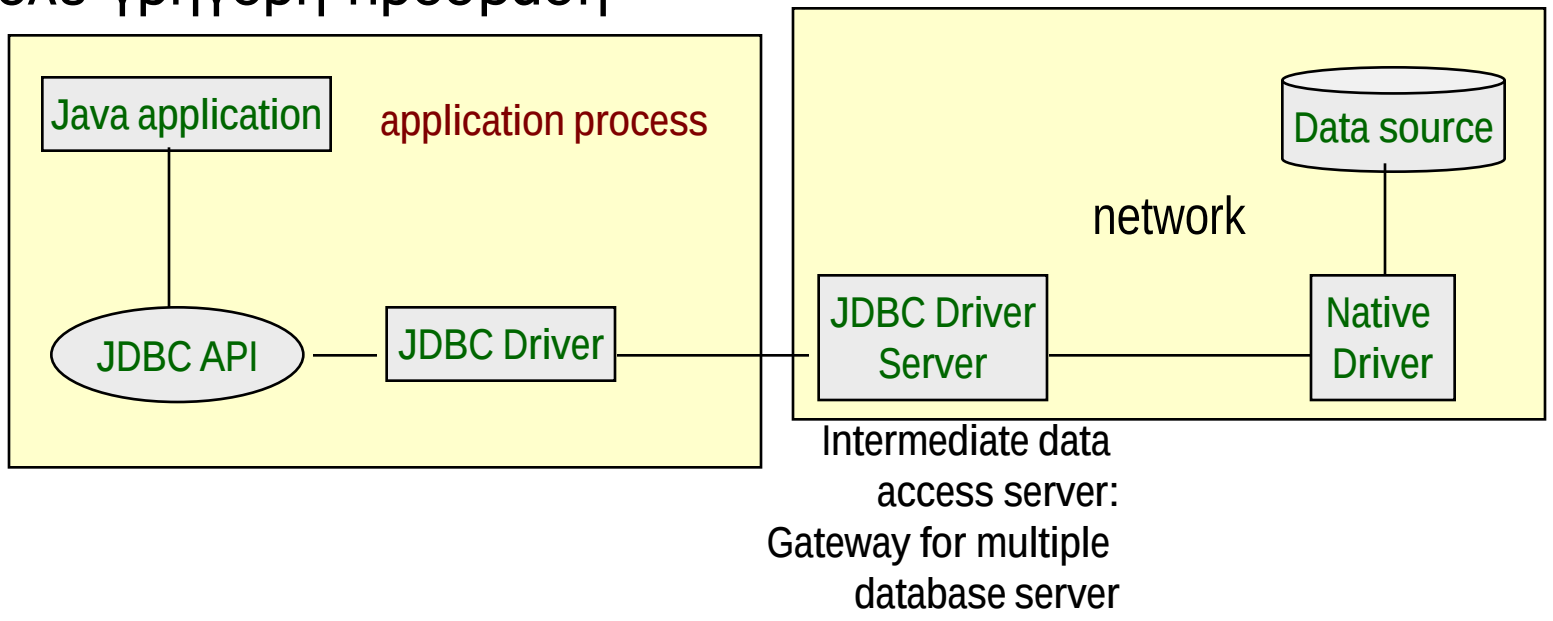
Μικτός κώδικας (Java και Native)

- Ο οδηγός JDBC έχει κώδικα java αλλά και κώδικα σε C (συνήθως) και μιλά απευθείας με το ΣΔΒΔ
- Πρέπει όποιος θέλει να συνδεθεί με τη ΒΔ να κατεβάσει το σωστό JDBC οδηγό
- Πιο αποτελεσματικό και γρήγορο από το JDBC-ODBC bridge



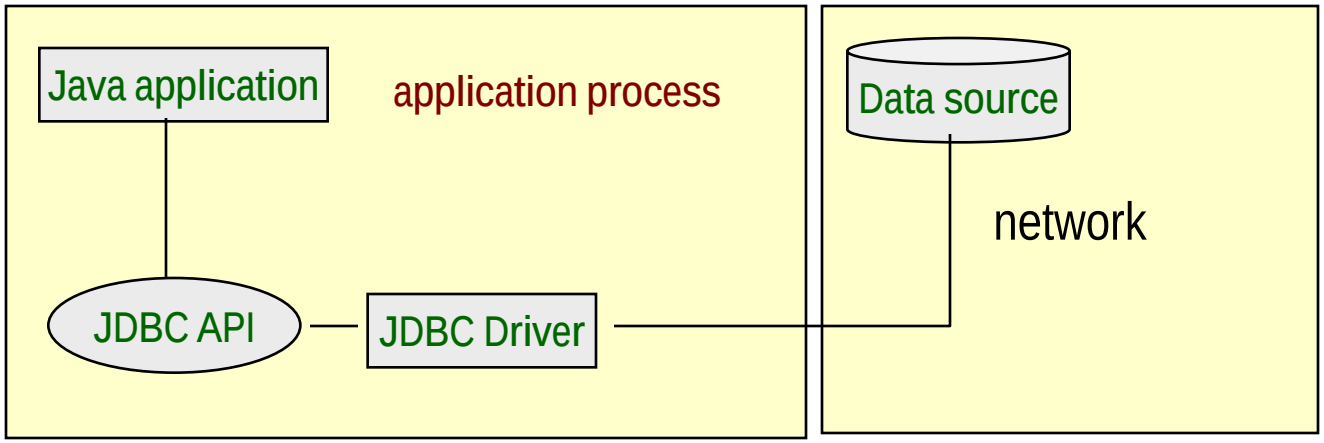
Χρήση ενδιάμεσου Server για την πρόσβαση

- Ένας ξεχωριστός server δίνει πρόσβαση στο ΣΔΒΔ. Ιδανική λύση για εσωτερικά δίκτυα (intranets).
- Οι δημιουργοί των ΣΔΒΔ παρέχουν τους επιπλέον servers που μιλούν με τις βάσεις τους και καταλαβαίνουν κλήσεις σε JDBC
- Ο οδηγός JDBC στον client μιλά με το JDBC server με κάποιο βασικό πρωτόκολλο
- Ο JDBC server μιλά με χρήση native οδηγού με τη βάση.
- Πολύ γρήγορη πρόσβαση



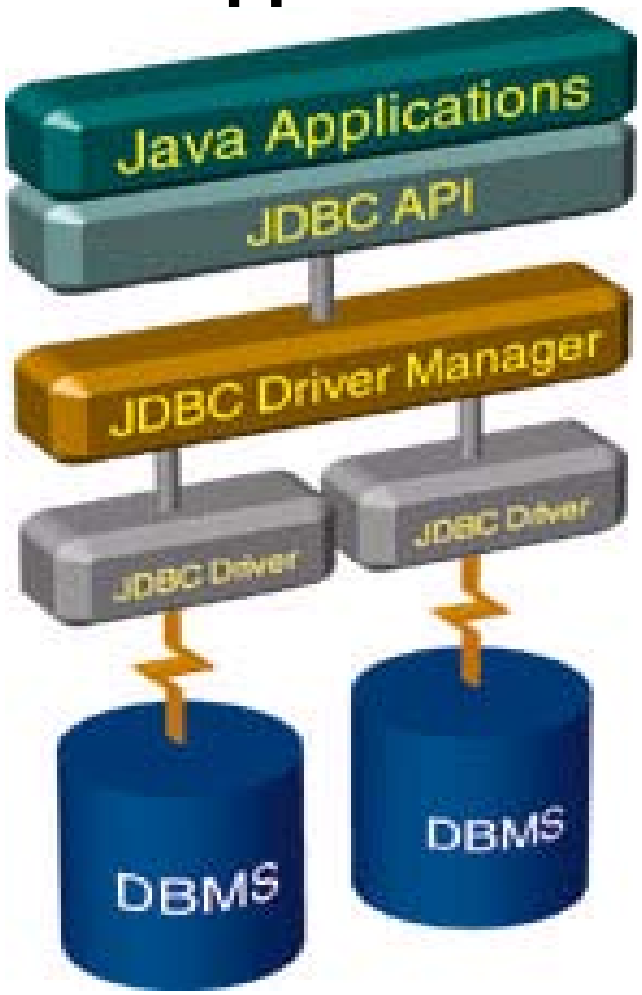
Καθαρός οδηγός Java

- Οι κλήσεις στον οδηγό JDBC μεταφράζονται σε κλήσεις στη βάση
- Απαιτεί από τον client να φορτώσει τον κατάλληλο οδηγό.
- Απλή αρχιτεκτονική και γρήγορη
- Οι δημιουργοί των ΣΔΒΔ παρέχουν τους pure java drivers



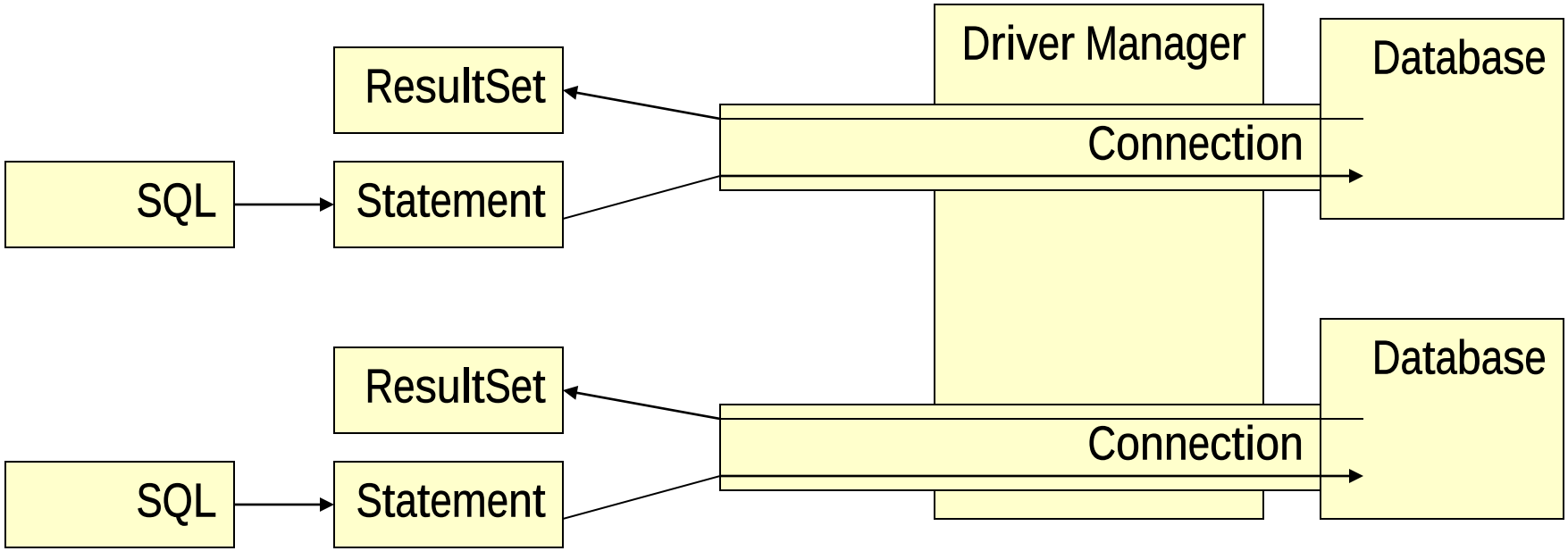
Ο διαχειριστής των οδηγών

- Μια εφαρμογή μιλά ταυτόχρονα σε πολλές ΒΔ
- Με διαφορετικό τύπο οδηγού στην κάθε μια
- Για το λόγο αυτό υπάρχει ο JDBC Driver Manager
- Ο DriverManager είναι μια τάξη που περιέχεται στο βασικό πακέτο java.sql
- Έχει τη μέθοδο getConnection για να δημιουργούμε μια σύνδεση σε μια ΒΔ



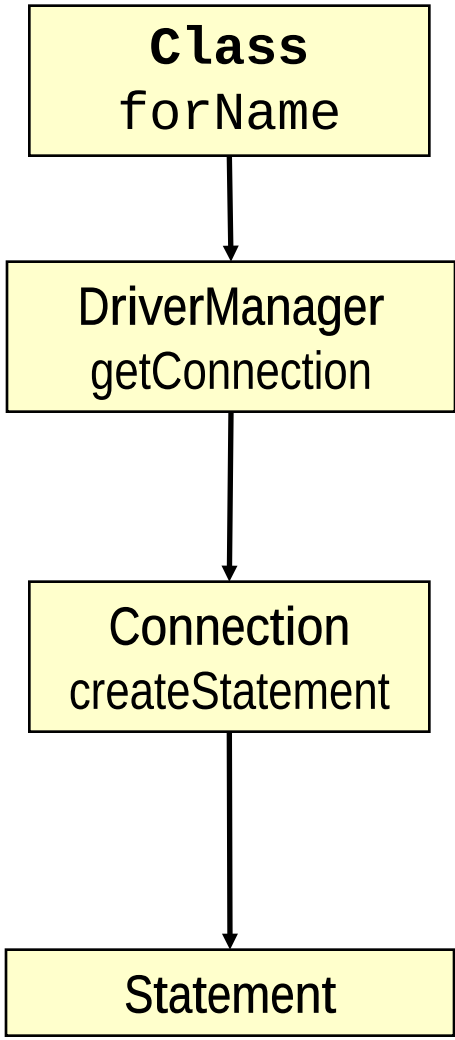
Το μοντέλο του JDBC – Τάξεις

- ***DriverManager***, διαχειρίζεται συνδέσεις σε ένα ή περισσότερα ΣΔΒΔ
- ***Connection,*** προσδιορίζει μία σύνδεση σε μια συγκεκριμένη ΒΔ
- ***Statement,*** περιέχει μια ερώτηση SQL που στέλνεται για εκτέλεση στη ΒΔ
- ***ResultSet,*** περιέχει τα αποτελέσματα εκτέλεσης της ερώτησης SQL (έχει τη μορφή πίνακα με μία ή περισσότερες εγγραφές-records)



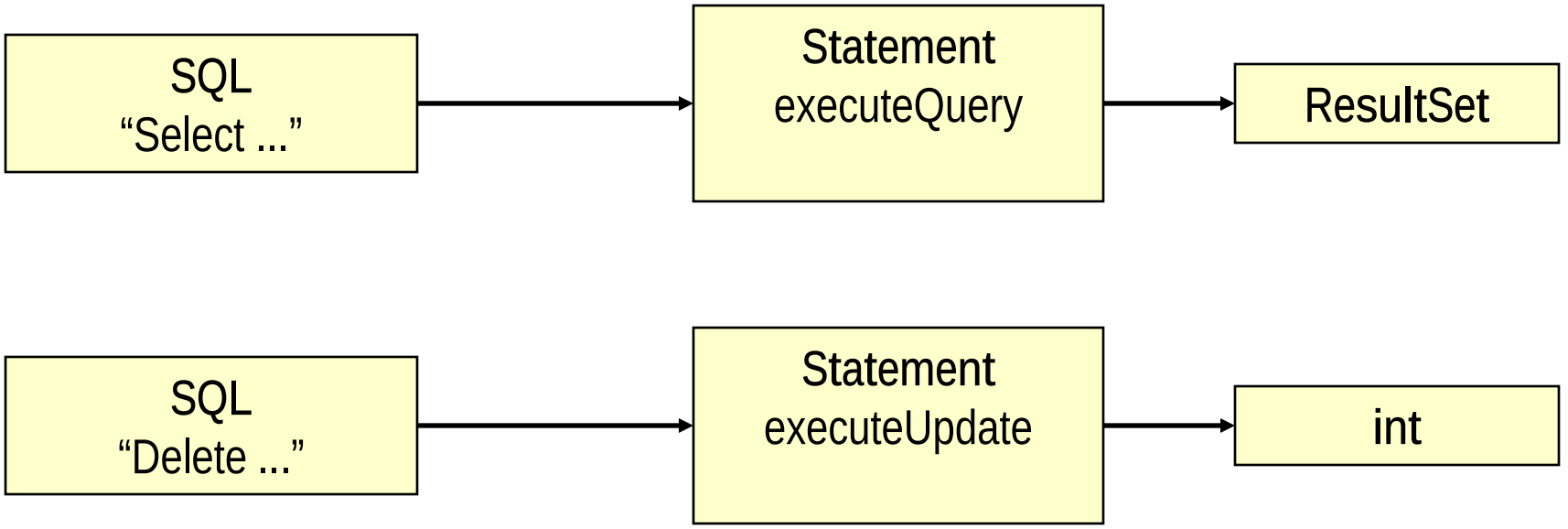
Βήματα χρήσης JDBC

1. Φορτώνουμε τον Driver.
2. Συνδεόμαστε με τη ΒΔ. Φτιάχνουμε ένα αντικείμενο *Connection*
3. Φτιάχνουμε μια δήλωση SQL (Statement) προς τη σύνδεση αυτή.
4. Εκτελούμε διαδοχικά (μια ή περισσότερες) ερωτήσεις SQL στο αντικείμενο Statement. Με χρήση της *executeQuery* ή *executeUpdate*
5. Επεξεργαζόμαστε το αντικείμενο ResultSet που φτιάχνεται όταν κάνουμε ερωτήσεις επιλογής στο βήμα 4
6. Κλείνουμε τα *Statement* και *Connection* αντικείμενα.



Βήματα χρήσης JDBC

Σε ένα αντικείμενο Statement στέλνουμε SQL ερωτήσεις και ενημερώσεις προς τη βάση. Οι ερωτήσεις επιστρέφουν είτε ένα αντικείμενο *ResultSet* είτε το πλήθος εγγραφών που ενημερώθηκαν.



Παράδειγμα

- Έστω ότι έχω εγκαταστήσει το ΣΔΒΔ Oracle στο μηχάνημα με IP διεύθυνση 10.100.51.123
- Η Oracle δέχεται κλήσεις στο port 1521
- Στο ΣΔΒΔ έχω φτιάξει μια δική μου βάση δεδομένων που λέγεται test
- Για να συνδεθώ στη βάση χρησιμοποιώ όνομα/κωδικό: test/test
- Στη βάση αυτή υπάρχει ο πίνακα Film που εμφάνισα σε προηγούμενη διαφάνεια.

1. Σύνδεση με τη βάση

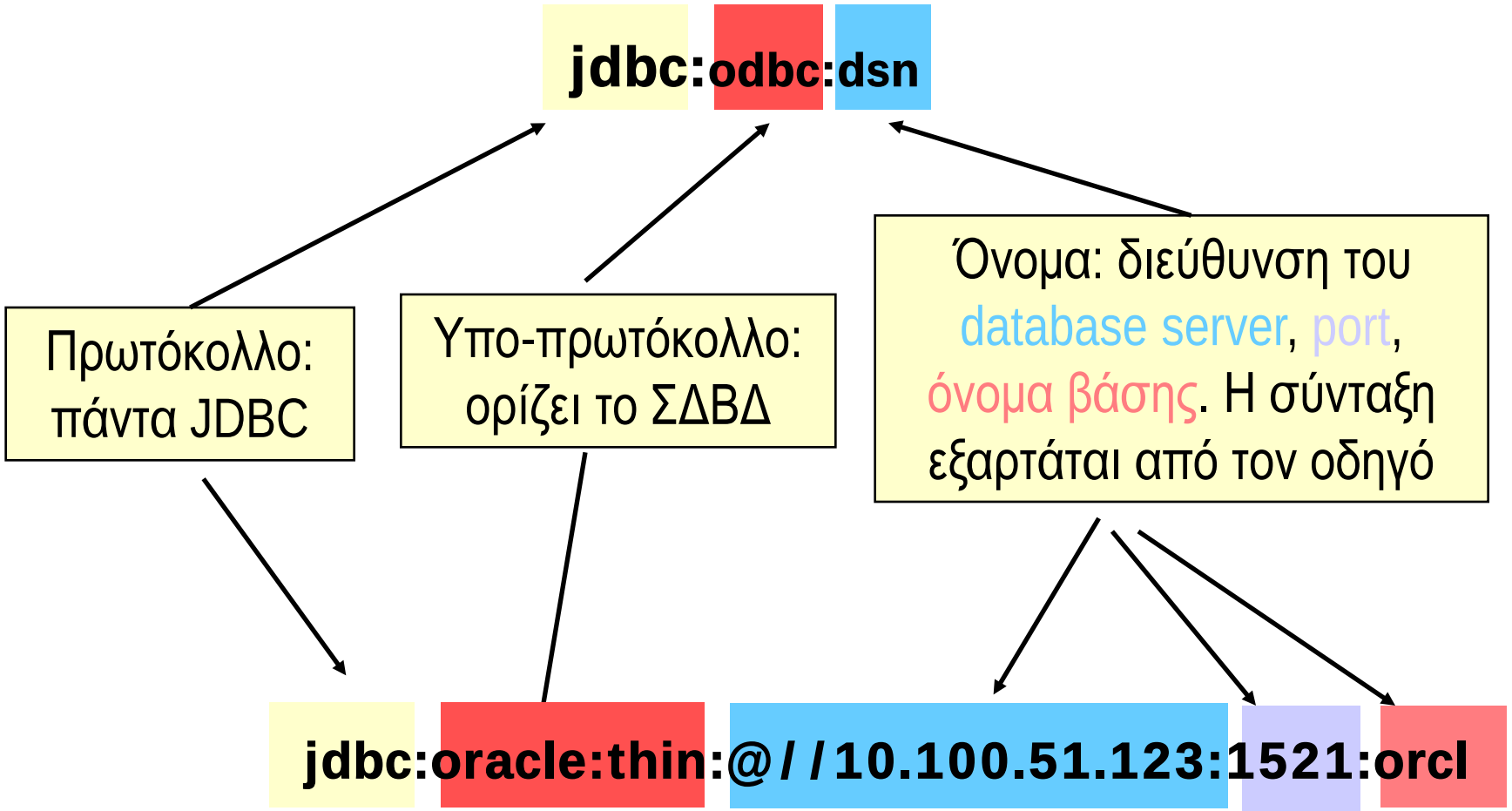
- **Jdbc:<subprotocol>:<subname>**

```
try {
    Class.forName("oracle.jdbc.driver.OracleDriver");
    String url ="jdbc:oracle:thin:@//10.100.51.123:1521/orcl";
Connection con = DriverManager.getConnection(url,“test", “test");
...
}
```

Πρέπει ο client να έχει τον αντίστοιχο οδηγό

- **Αν έχω Windows και θέλω να χρησιμοποιήσω τον πρώτο τύπο driver, φτιάχνω ένα νέο όνομα πηγής δεδομένων στο OBDC των Windows (έστω με όνομα dsn) και φορτώνω τον αντίστοιχο οδηγό.**

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
String url ="jdbc:odbc:dsn";
```


jdbc:odbc:dsn
Πρωτόκολλο: πάντα JDBC
Υπο-πρωτόκολλο: ορίζει το ΣΔΒΔ
Όνομα: διεύθυνση του database server, port, όνομα βάσης. Η σύνταξη εξαρτάται από τον οδηγό
jdbc:oracle:thin:@//10.100.51.123:1521:orcl

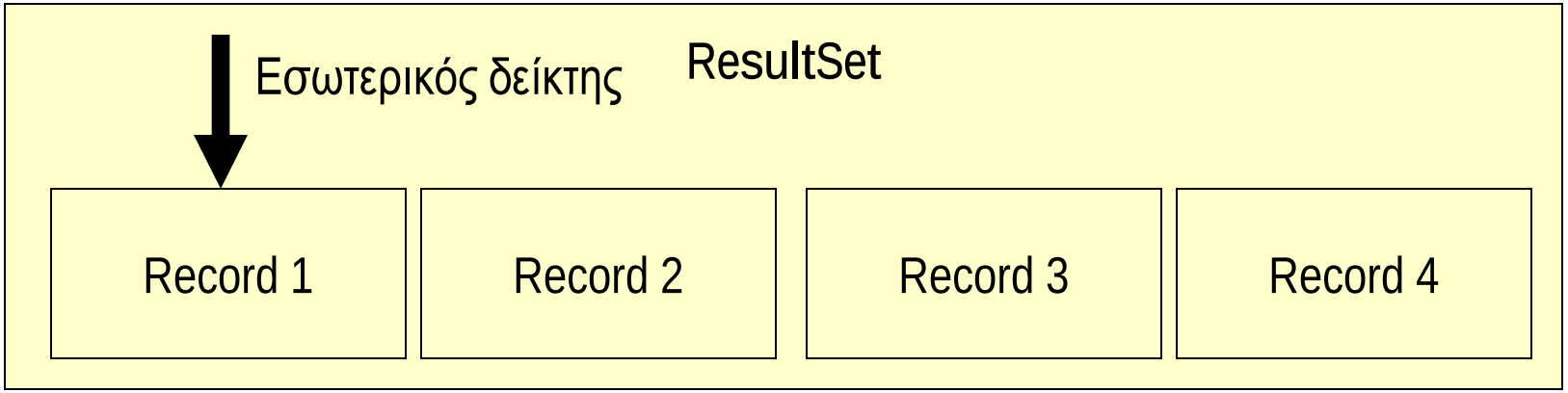
2. Διαμόρφωση SQL ερωτήσεων

```
try{...
Statement stmt = con.createStatement();
String sqlselect = "SELECT Title, Director,Year FROM
    Film";
ResultSet rs = stmt.executeQuery(sqlselect);
int rowschanged = stmt.executeUpdate("UPDATE Film
    set Year=1971 WHERE Title='Birds' ");
...}
```

- Όλη η επικοινωνία με τη ΒΔ γίνεται μέσα σε ένα βρόχο try που ακολουθείται από διαδικασίες ανίχνευσης λάθους (SQLException κλπ.)

Διαχείριση αποτελεσμάτων

- Το ResultSet έχει ένα εσωτερικό δείκτη που μπορεί να μετακινηθεί μπρος/πίσω.
- Κάθε φορά μπορούμε να πάρουμε τις τιμές της τρέχουσας εγγραφής
  - Μέθοδοι: next(), previous(), first(), last(). Μετακινούν το δείκτη και επιστρέφουν true αν υπάρχει εγγραφή εκεί που πάει ο δείκτης. π.χ. αν ο δείκτης είναι στην αρχή η Previous() επιστρέφει false κ.ο.κ.
  - Οι isLast() και isFirst() ελέγχουν την τρέχουσα θέση του δείκτη



3. Εμφάνιση αποτελεσμάτων

```
try {…
    ResultSet rs = stmt.executeQuery(sqlselect);
    while(rs.next()){
       title = rs.getString(1);
       director = rs.getString(2);
       year = rs.getInt(3);
       row+=title+” “ +director+” “+year+”\n”;
    }
    stmt.close();  //κλείσιμο της δήλωσης
    con.close();   //κλείσιμο της σύνδεσης
}
catch(Exception e) {…}
```

Ακύρωση και υποβολή ερώτησης

- Εξ ορισμού κάθε ερώτηση που γίνεται στη ΒΔ υποβάλλεται (γίνεται και commit)
- Αν θέλουμε να κάνουμε πολλαπλές ερωτήσεις/ενημερώσεις και στο τέλος να υποβάλλουμε τα αποτελέσματα:
- απενεργοποιούμε την αυτόματη υποβολή.

  *connection.**setAutoCommit**(false)*
- και καλούμε

  *connection.**commit**()* *για να εφαρμόσουμε τις αλλαγές*

  *connection.**rollback**()* για να ακυρώσουμε τις αλλαγές ως την τελευταία εφαρμογή

Τεχνικές πρόσβασης στη ΒΔ

Παράδειγμα: JDBC

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
        //... try {
                    Connection conn =
        DriverManager.getConnection("jdbc:mysql://localhost/test?user=monty&password=some");
        // Do something with the Connection // ...
        } catch (SQLException ex) {
/       / handle any errors
        System.out.println("SQLException: " + ex.getMessage()); System.out.println("SQLState: " +
ex.getSQLState()); System.out.println("VendorError: " + ex.getErrorCode());
//...
```

MySQL Connector/J Documentation

Παράδειγμα: JDBC (συνέχ.)

```
ResultSet rs = null;
try { stmt = conn.createStatement();
rs = stmt.executeQuery("SELECT foo FROM bar");
// or alternatively, if you don't know ahead of time that // the query will be a SELECT...
if (stmt.execute("SELECT foo FROM bar")) {
        rs = stmt.getResultSet(); }
       // Now do something with the ResultSet ....
} catch (SQLException sqe) {
// it is a good idea to release
}
```

Πλαίσια απεικόνισης αντικειμένων σε σχεσιακό

- Τι προσφέρουν
  - Διαφανή υποστήριξη «παραμενόντων αντικειμένων» (transparent persistence)
  - Caching
  - Διαχείριση δοσοληψιών (Transaction management)
- Που θα βρω πληροφορίες
  - https://docs.oracle.com/javaee/7/tutorial/partpersist.htm#BNBPY

Ένα παράδειγμα με το Hibernate

- Δύο κλάσεις του πεδίου εφαρμογής:
  - Book
  - Author
  - Με σχέση πολλά προς πολλά.

Η κλάση Book

```
package example;
@Entity
public class Book {

  @Id
  @GeneratedValue
  private Integer id;
  private String title;
  private String subtitle;

  @ManyToMany
  private Set<Author> authors = new HashSet<Author>();

  private Date publicationDate;

  public Book() {
  }

  // standard getters/setters follow here
  ...
}
```

Η κλάση Author

```
package example;
@Entity
public class Author {

  @Id
  @GeneratedValue
  private Integer id;
  private String name;
  public Author() {
  }
   // standard getters/setters follow here
  ...
}
```

Η κλάση Book Manager

```
class BookManager {
private void createAndStoreBook(String title, Date theDate) {
        Session session = HibernateUtil.currentSession();
        Transaction tx = session.beginTransaction();

        Book aBook = new Book();
        aBook.setTitle(“Introduction to Software Engineering”);
        //...
        session.save(theBook);
        tx.commit();
        HibernateUtil.closeSession();
    }
}
```