Βάσεις Δεδομένων

Εισαγωγή στην διασύνδεση βάσεων δεδομένων με την πλατφόρμα Java (JDBC-Java Database Connectivity)

Εργαστήριο Διανεμημένων Πληροφοριακών Συστημάτων & Εφαρμογών

Database Connectivity Evolution

- Πρώιμα Χρόνια (1960s - 1970s)

  Άμεση Πρόσβαση (Direct Access)
  - Οι εφαρμογές επικοινωνούσαν απευθείας με τη βάση δεδομένων μέσω ειδικών (proprietary) APIs που έδινε κάθε κατασκευαστής.
  - Κάθε βάση είχε το δικό της σύστημα επικοινωνίας
- Εμφάνιση της SQL (1970s)
  - Η SQL εμφανίστηκε το 1974 από την IBM.
  - Τυποποιημένη γλώσσα επικοινωνίας με βάσεις δεδομένων.
  - Όμως, οι εφαρμογές συνέχιζαν να χρησιμοποιούν APIs συγκεκριμένα για κάθε βάση
- ODBC (Αρχές 1990)

  Open Database Connectivity
  - Δημιουργήθηκε από τη Microsoft το 1992.
  - Παρείχε ένα κοινό API επικοινωνίας με οποιαδήποτε SQL βάση. Υλοποίηση σε C.

Database Connectivity Evolution (2)

- JDBC (Μέσα 1990)

  Java Database Connectivity
  - Εμφανίστηκε το 1996 από την Sun Microsystems.
  - Ειδικά σχεδιασμένο για Java.
  - Πλήρως ανεξάρτητο από πλατφόρμα (pure Java).

- Σύγχρονες Τεχνολογίες (2000s - Σήμερα)
  - ORM Frameworks (που κρύβουν τη SQL)
    Hibernate, JPA, Spring Data

- Οι εφαρμογές επικοινωνούν με REST APIs ή GraphQL

Database Connectivity Evolution (2)

- Architecture Evolution

1. Early:
App --> Proprietary API --> Database

2. ODBC:
App --> ODBC API --> ODBC Driver --> Database

3. JDBC:
Java App --> JDBC API --> JDBC Driver --> Database

4. ORM:
Java App --> ORM Framework --> JDBC --> Database

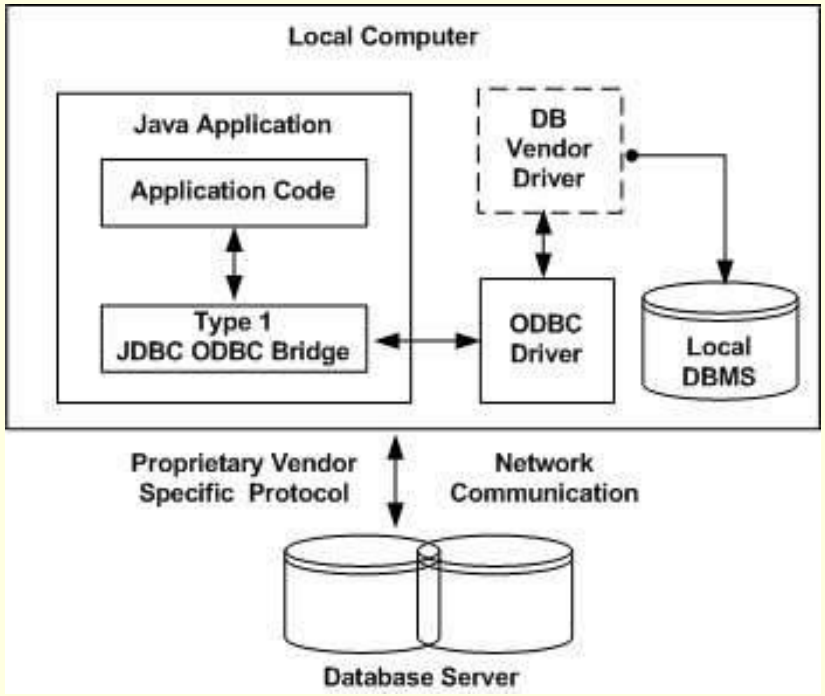
5. Modern:
App --> REST API / GraphQL --> Backend (Java) --> JDBC --> Database

Database Connectivity - JDBC

- Το JDBC παρέχει μια πρότυπη βιβλιοθήκη για την πρόσβαση σε συστήματα διαχείρισης σχεσιακών βάσεων δεδομένων μέσω της πλατφόρμας Java
  - Το JDBC-API παρέχει πρότυπες κλάσεις και διεπαφές για:
    - Την σύνδεση σε βάσεις δεδομένων
    - Την αρχικοποίηση ερωτήσεων
    - Την δημιουργία αποθηκεύσιμων ερωτήσεων
    - Την δομή δεδομένων για το αποτέλεσμα μιας ερώτησης
    - Την περιγραφή μετα-δεδομένων
  - Το JDBC-API δεν καθορίζει την σύνταξη της SQL
  - JDBC δεν είναι embedded SQL

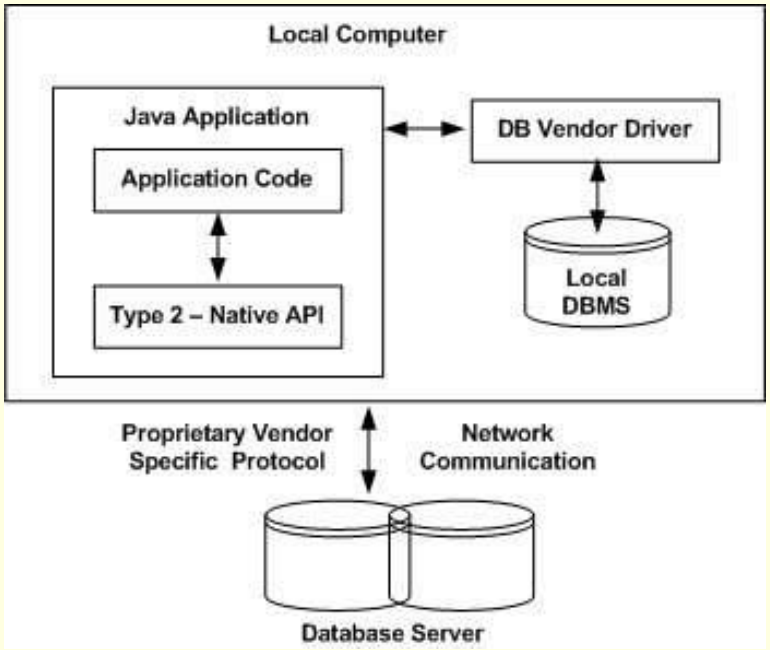
Τύποι οδηγών τεχνολογίας JDBC

- ***JDBC-ODBC bridge:*** Αυτός ο τύπος οδηγού προσφέρει πρόσβαση σε βάσεις δεδομένων μέσω ενός η περισσοτέρων οδηγών ODBC. Σημειώστε ότι εδώ χρειάζεται να φορτωθεί δυναμικά στην πλευρά του πελάτη ο απαραίτητητος κώδικας που υλοποιεί το πρωτόκολλο ODBC. Ένας οδηγός JDBC-ODBC bridge παρέχεται ελεύθερα από την Oracle



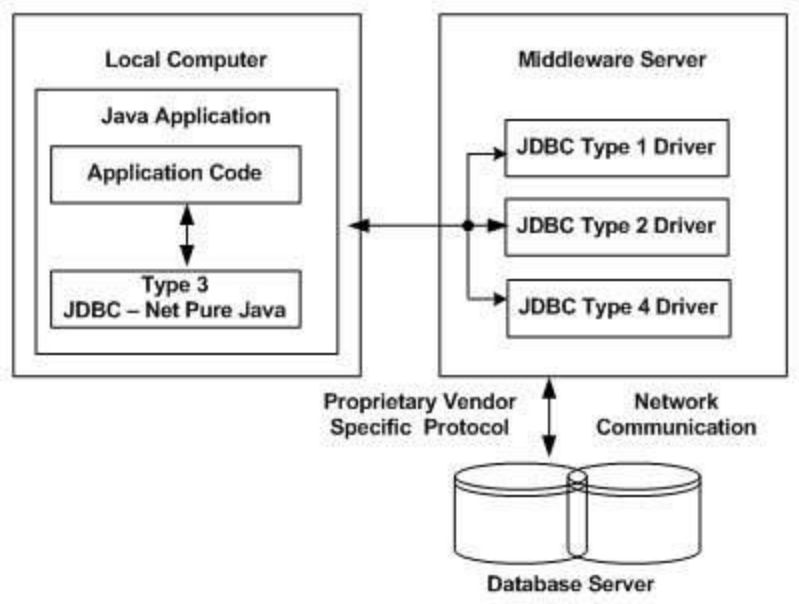
Τύποι οδηγών τεχνολογίας JDBC

- ***Native-API partly Java technology-enabled driver:*** Αυτός ο τύπος οδηγού μετατρέπει τις εντολές JDBC σε εντολές που δέχονται συγκεκριμένα συστήματα (client API για Oracle, Sybase, Informix, DB2). Σημειώστε ότι εδώ χρειάζεται να φορτωθεί δυναμικά στην πλευρά του πελάτη ο απαραίτητητος κώδικας για την επικοινωνία με τα συστήματα βάσεων δεδομένων



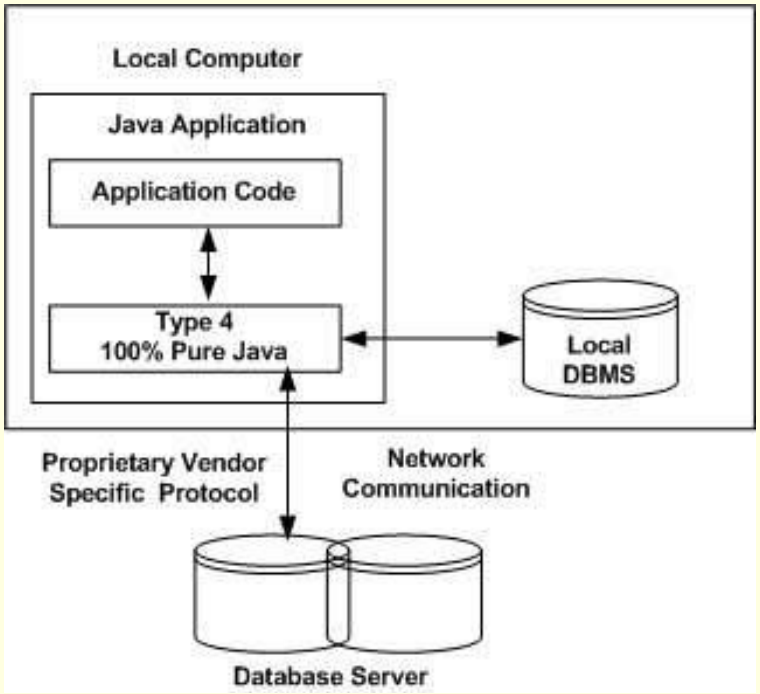
Τύποι οδηγών τεχνολογίας JDBC

- ***Net-protocol fully Java technology-enabled driver:*** Αυτός ο τύπος οδηγού μετατρέπει τις εντολές JDBC σε εντολές που ακολοθούν ένα πρωτόκολλο δικτύου ανεξάρτητο από συστήματα βάσεων δεδομένων και στην συνέχεια αυτές οι εντολές μετατρέποντια κατάλληλα στο πρωτόκολλο που χρησιμοποιεί το σύστημα διαχείρισης βάσεων δεδομένων. Αυτό το λογισμικό δικτύου μεσαίου επιπέδου (net middleware) μπορεί να διασυνδέσει όλες τις Java εφαρμογές με ένα πληθος από διαφορετικές βάσεις δεδομένων.



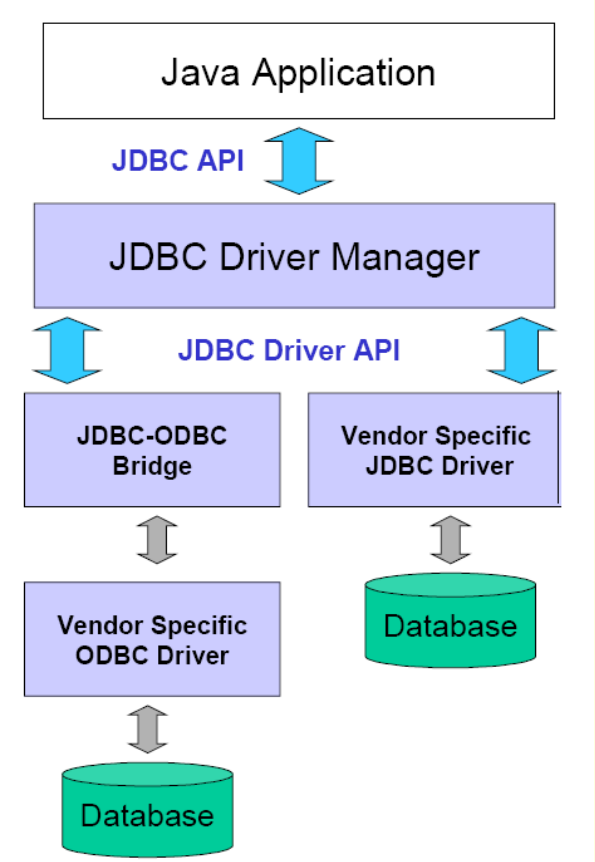
Τύποι οδηγών τεχνολογίας JDBC

- ***Native-protocol fully Java technology-enabled driver:*** Αυτός ο τύπος οδηγού μετατρέπει τις εντολές JDBC σε εντολές δικτύου που χρησιμοποιούνται απευθείας από το σύστημα διαχείρισης βάσεων δεδομένων. Με αυτό τον τρόπο επιτρέπεται η απευθείας κλήση από το μηχάνημα του πελάτη στον εξυπηρετητή βάσεων δεδομένων και αποτελεί πρακτική λύση για την πρόσβαση σε βάσεις δεδομένων μέσω intranet. Εφόσον αυτό το πρωτόκολλο απευθείας επικοινωνίας εξαρτάται από το σύστημα βάσης δεδομένων οι κατασκευαστές συστημάτων βάσεων δεδομένων συνήθως παρέχουν και του κατάλληλους JDBC οδηγούς για τα συστήματά τους



Εισαγωγή στην τεχνολογία JDBC

- Το πρότυπο JDBC αποτελείται απο δύο μέρη:
  - JDBC-API (Καθορισμός διαπεφών σε Java)
  - Διαχειριστής Οδηγών JDBC επικοινωνεί με οδηγούς διαφόρων κατασκευαστών/εταιρειών που αναλαμβάνουν την επικοινωνία με την βάση
    - Η «μετάφραση» της επικοινωνίας από την πρότυπη διεπαφή στην συγκεκριμένη τεχνολογική προσέγγιση μιας βάσης γίνεται στο επίπεδο του πελάτη

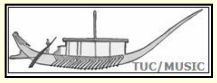


JDBC Components

- **DriverManager**: Αυτή η κλάση διαχειρίζεται μια λίστα προγραμμάτων οδήγησης βάσης δεδομένων.
- **Driver**: Χειρίζεται τις επικοινωνίες με το διακομιστή βάσης δεδομένων. Σπάνια θα αλληλεπιδράσετε με αντικείμενα του προγράμματος οδήγησης. Αντ 'αυτού, χρησιμοποιείτε τον DriverManager, ο οποίος διαχειρίζονται αντικείμενα αυτού του τύπου.
- **Connection**: Χρησιμοποιείται για την εκτέλεση όλων των μεθόδων επικοινωνίας σε μια βάση δεδομένων.
- **Statement**: Χρησιμοποιείται για να υποβάλετε SQL statements στη βάση δεδομένων.
- **ResultSet**: Διατηρεί δεδομένα που ανακτώνται από μια βάση δεδομένων μετά την εκτέλεση ενός SQL statement

Τύποι Δεδομένων - JDBC

QL	JDBC/Java	setXXX	getXXX
VARCHAR	**java.lang.String**	setString	getString
CHAR	**java.lang.String**	setString	getString
LONGVARCHAR	**java.lang.String**	setString	getString
BIT	**boolean**	setBoolean	getBoolean
NUMERIC	**java.math.BigDecimal**	setBigDecimal	getBigDecimal
TINYINT	**byte**	setByte	getByte
SMALLINT	**short**	setShort	getShort
INTEGER	**int**	setInt	getInt
BIGINT	**long**	setLong	getLong
REAL	**float**	setFloat	getFloat
FLOAT	**float**	setFloat	getFloat
DOUBLE	**double**	setDouble	getDouble
VARBINARY	**byte[ ]**	setBytes	getBytes
BINARY	**byte[ ]**	setBytes	getBytes
DATE	**java.sql.Date**	setDate	getDate
TIME	**java.sql.Time**	setTime	getTime
TIMESTAMP	**java.sql.Timestamp**	setTimestamp	getTimestamp
CLOB	**java.sql.Clob**	setClob	getClob
BLOB	**java.sql.Blob**	setBlob	getBlob
ARRAY	**java.sql.Array**	setARRAY	getARRAY



Βασικά βήματα χρήσης JDBC

1. Φόρτωση του οδηγού της βάσης
2. Ορισμός του URL της σύνδεσης
3. Εγκατάσταση της σύνδεσης
4. Δημιουργία του αντικειμένου Statement
5. Εκτέλεση της ερώτησης
6. Επεξεργασία των αποτελεσμάτων
7. Κλείσιμο της σύνδεσης

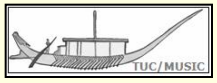
Βασικά βήματα χρήσης JDBC

1. Φόρτωση του οδηγού της βάσης

```
try {
        Class.forName("com.mysql.jdbc.Driver");
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Class.forName("org.postgresql.Driver");
} catch(Exception e) {
        e.printStackTrace();
 }
```

2. Καθορισμός του URL της σύνδεσης

```
String host = "test.ced.tuc.gr";
String dbName = "someName";
Int port = 3306;
String mysqlURL="jdbc:mysql://"+host+":"+port+"/"+dbName
---------------------------------------------------------------------------
String host = "test.ced.tuc.gr";
String dbName = "someName";
Int port = 5432;
String postgresURL="jdbc:postgresql://"+host+":"+port+"/"+dbName;
```



Βασικά βήματα χρήσης JDBC

3. Εγκατάσταση της σύνδεσης

```
String username = “nikos”;
String password = “secret”;
Connection conn = DriverManager.getConnection(mysqlURL,username,password);
```

- Προαιρετικά από την σύνδεση μπορούμε να πάρουμε διάφορες πληροφορίες

```
DatabaseMetaData dbMetadata = conn.getMetaData();
String productName = dbMetadata.getDatabaseProductName();
System.out.println("Database: "+ productName);
String productVersion = dbMetadata.getDatabaseProductVersion();
System.out.println("Version: "+ productVersion);
```

Βασικά βήματα χρήσης JDBC

4. Δημιουργία του αντικειμένου Statement

```
Statement statement = conn.createStatement();
```

5. Εκτέλεση της ερώτησης

```
String query = "SELECT col1, col2, col3 FROM sometable";
ResultSet resultSet = statement.executeQuery(query);
```

- Για να ενημερώσουμε μια βάση δεδομένων χρησιμοποιούμε executeUpdate() ενώ στην ερώτηση εμπεριέχονται οι αντίστοιχες λέξεις κλειδιά «UPDATE, INSERT , DELETE»
- Η κλήση της setQueryTimeout() καθορίζει τον μέγιστο χρόνο που μπορεί να περιμένει η εφαρμογή για την εκτέλεση μιας ερώτησης

Βασικά βήματα χρήσης JDBC

6. Επεξεργασία των αποτελεσμάτων

```
while (resultSet.next()) {
        System.out.println(resultSet.getInt(1)+" "+resultSet.getString(2));
}
```

- Η πρώτη στήλη έχει δείκτη 1 και όχι 0
- Το αντικείμενο ResultSet παρέχει διάφορες μεθόδους getXxxx() οι οποίες παίρνουν σαν όρισμα τον δείκτη της στήλης ή το όνομα του γνωρίσματος και επιστρέφουν τα αντίστοιχα δεδομένα

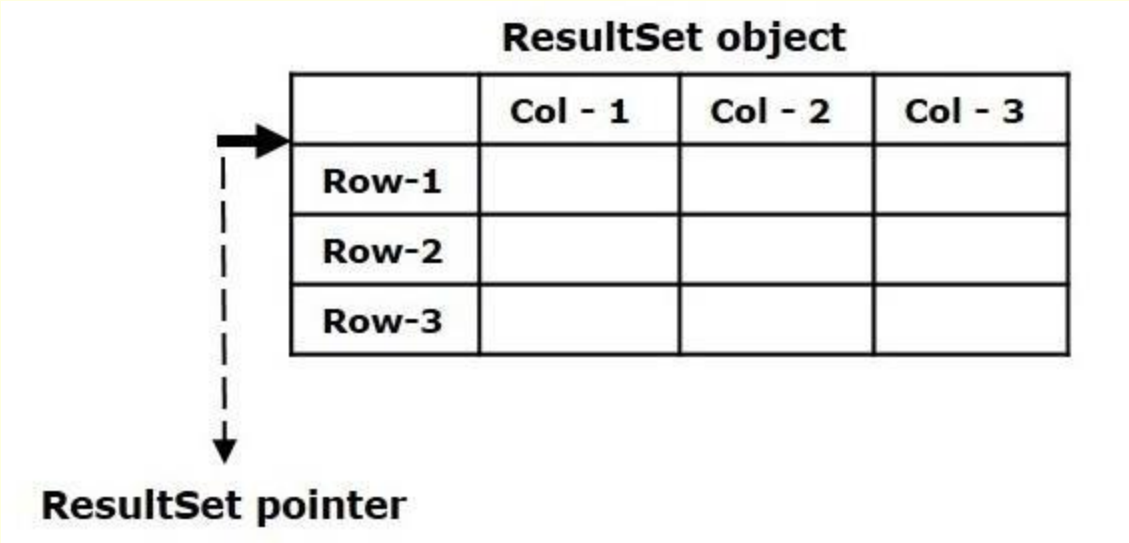
7. Κλείσιμο της σύνδεσης

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

- Η εγκατάσταση μιας νέας σύνδεσης είναι σχετικά «ακριβή» διαδικασία. Για αυτόν τον λόγο συνήθως δεν κλείνουμε την σύνδεση αν πρόκειται να συμβουν πολλές λειτουργίες στην βάση.

ResultSet

- Ένα αντικείμενο ResultSet περιέχει τα αποτελέσματα μιας SQL ερώτησης.
- Αναπαριστάται σαν ένας πίνακας με γραμμές και στήλες



ResultSet (2)

Type
ResultSet.TYPE_FORWARD_ONLY
ResultSet.TYPE_SCROLL_INSENSITIVE
ResultSet.TYPE_SCROLL_SENSITIVE.

Concurrency
ResultSet.CONCUR_READ_ONLY
ResultSet.CONCUR_UPDATABLE

Statement

- SQL εντολές στέλνονται στο σύστημα βάσεων δεδομένων μέσω του αντικειμένου Statement
- Τρεις τύποι αντικειμένων Statement είναι διαθέσιμοι
  - Statement
    - Για την εκτέλεση απλών εντολών σε SQL
  - PreparedStatement
    - Για την εκτέλεση εντολών SQL που έχουν προ-επεξεργαστεί και παραμετροποιηθεί
  - CallableStatement
    - Για την εκτέλεση stored procedures

Prepared Statement

- Αν η εφαρμογή μας πρόκειται να εκτελεί συχνά παρόμοιες SQL ερωτήσεις η χρήση της κλάσης PreparedStatement είναι πιο αποδοτική
- Επιτρέπει την δημιουργία ενός statement με καθορισμένο τρόπο το οποίο στέλνεται στο σύστημα βάσεων δεδομένων για προ-επεξεργασία και μετάφραση (compilation) πριν αρχίσει να χρησιμοποιείται
- Κάθε φορά που χρησιμοποιείται απλά εισάγονται τιμές στις παραμέτρους με την κλήση των μεθόδων setXxx()
- Μέθοδοι
  - executeQuery()
  - executeUpdate()

Callable Statement

- Επιτρέπει την κλήση stored procedures στην βάση δεδομένων
- Ο προγραμματιστής χρειάζεται να γνωρίζει μόνο της παραμέτρους εισόδου και εξόδου και όχι το σχήμα της βάσης ή την εσωτερική υλοποιήση της stored procedure
- Επιτρέφει εγγραφή παραμέτρων εξόδου και ανάκτηση των τιμών τους.

Διαχείριση Εξαιρέσεων

- SQL Exceptions
  - Σχεδόν κάθε JDBC μέθοδος μπορεί να δημιουργήσει μια εξαίρεση σαν απάντηση σε κάποιο λάθος στην βάση δεδομένων
  - Αν προκύψουν περισσότερα από ένα λάθη οι εξαιρέσεις ενώνονται μεταξύ τους σε αλυσίδα
  - Οι SQL εξαιρέσεις (SQL Exceptions) περιέχουν:
    - Περιγραφή του λάθους (getSQLState)
    - Ένα ακέραιο αριθμό λάθους διαφορετικό για κάθε κατασκευαστή (getErrorCode)
    - Σύνδεσμο στην επόμενη εξαίρεση (getNextException)

MetaData

- Τα αντικείμενα DatabaseMetadata και ResultSetMetaData μας δίνουν χρήσιμες πληροφορίες όπως:
  - Σε ποιο σύστημα βάσεων δεδομένων έχουμε συνδεθεί και ποια εκδοση
  - Πόσες στήλες υπάρχουν στο ResultSet
  - Ποιό είναι το όνομα μιας συγκεκριμένης στήλης
  - Ποιος είναι ο τύπος δεδομένων μιας συγκεκριμένης στήλης

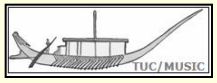
MetaData (2)

- Παράδειγμα

```
Class.forName(dbDriver);
Connection conn = DriverManager.getConnection(dbString, dbUsername, dbPassword);
System.out.println(conn.getMetaData().getDatabaseProductVersion());
DatabaseMetaData dbMetadata = conn.getMetaData();
String productName = dbMetadata.getDatabaseProductName();
System.out.println("Database: "+ productName);
String productVersion = dbMetadata.getDatabaseProductVersion();
System.out.println("Version: "+ productVersion);

Statement statement = conn.createStatement();
String query = "SELECT * FROM orders_t";
ResultSet resultSet = statement.executeQuery(query);

ResultSetMetaData rsm = resultSet.getMetaData();
int columnCount = rsm.getColumnCount();
for (int i=1;i<columnCount;i++)
   System.out.print(rsm.getColumnName(i)+" ");

while (resultSet.next())  {
   System.out.println("");
   for (int i = 1; i < columnCount; i++)
      System.out.print(resultSet.getString(1) + " ");
}
```

ΑΠΟΤΕΛΕΣΜΑ

Database: MySQL
Version: 5.0.37-community-nt
order_id cust_id product_id quantity price
1 4 5 10 100
2 2 7 8 120



Παραδείγματα Χρήσης Τεχνολογίας JDBC

- Σύνδεση με χρήση οδηγού JDBC
- Χρήση Statement για εκτέλεση απλών ερωτήσεων
- Χρήση PreparedStatement για εκτέλεση παραμετροποιημένων, προ-επεξεργασμένων ερωτήσεων
- Χρήση CallableStatement για κλήση stored procedures

JDBC Example : Statement

```
public static void main(String[] args) {
    try {
      DBC db= new DBC();
      Statement st = db._conn.createStatement();
      ResultSet rs =  st.executeQuery("select movieID, title from Movie as m, Title as t"+
                                          "where m.titleID=t.titleID");
      while (rs.next()) {
        System.out.println("result: "+rs.getInt(1)+"  "+rs.getString(2));
      }
      rs.close();
      st.close();
    }catch (Exception ex) {
      ex.printStackTrace();
    }
}
```

JDBC Example: PreparedStatement

```
public static void main(String[] args) {
    try {
      DBC db= new DBC();
      PreparedStatement pst
                        = db._conn.prepareStatement("select * from Movie where movieID=?");
      pst.setInt(1,1);
      ResultSet rs = pst.executeQuery();
      while (rs.next()) {
        System.out.println("result: "+rs.getInt(1)+"  "+rs.getInt(2));
      }
      rs.close();
      pst.close();
    }catch (Exception ex) {
      ex.printStackTrace();
    }
  }
```

JDBC Example: CallableStatement

```
public static void main(String[] args) {
    try {
      DBC db= new DBC();

      CallableStatement cst = db._conn.prepareCall(“call from showMovie(?)");
      cst.setInt(1,1);
      ResultSet rs = cst.executeQuery();
      while (rs.next()) {
        System.out.println("result: "+rs.getInt(1)+"  "+rs.getString(2));
      }
      rs.close();
      cst.close();
    }catch (Exception ex) {
      ex.printStackTrace();
    }
  }
```

JDBC Example: CallableStatement

```
public static void main(String[] args) {
    try {
      DBC db= new DBC();
      CallableStatement cst = db._conn.prepareCall(“call showMovieNo(?) ");
      cst.registerOutParameter(1,java.sql.Types.INTEGER);
      cst.executeUpdate();
      System.out.println("Total No of movies = "+cst.getInt(1));
      cst.close();
    }catch (Exception ex) {
      ex.printStackTrace();
    }
}
```

Postgres JDBC Driver

- Documentation
  - http://jdbc.postgresql.org