JDBC

Jeszenszky Péter
Debreceni Egyetem, Informatikai Kar
jeszenszky.peter@inf.unideb.hu

Utolsó módosítás: 2018. december 20.

JDBC

- Relációs adatok elérésére szolgáló Java API.
  - Lehetővé teszi SQL lekérdezések végrehajtását, az eredmények beolvasását és az alapul szolgáló adatforrás módosítását.
  - Leképezés megvalósítása a Java programozási nyelv és az SQL között.
- A Java SE és a Java EE része.
  - A `java.sql` és a `javax.sql` csomagok tartalmazzák.
    - A csomagokat a JDK 9 `java.sql` modulja tartalmazza.

Tervezési célok

- Illeszkedjen a Java SE és Java EE platformokba.
- Gyártófüggetlen módon tegye lehetővé az implementációk által elterjedten támogatott lehetőségek használatát.
- Magasabb szintű eszközök és API-k alapjául szolgálhasson.
- Egyszerűség.

Specifikáció

- *JSR 221: JDBC 4.3 API Specification* (July 2017) https://jcp.org/en/jsr/detail?id=221
  - Támogatott SQL szabvány: SQL:2003

Történet (1)

- A JDK 1.1-ben jelent meg 1997-ben.
- Az előző verzió a 4.2 számú, melyet a Java SE 8 tartalmaz.
  - Újdonságok:
    - Új interfészek és osztályok: `java.sql.SQLType`, `java.sql.DriverAction`, `java.sql.JDBCType.`
    - A `java.time` csomag dátum és idő osztályainak támogatása.
    - …

Történet (2)

- A jelenleg aktuális verzió a 4.3 számú, melyet a Java SE 9 tartalmaz.
  - Újdonságok:
    - *Sharding* támogatás.
    - Új interfészek: `java.sql.ConnectionBuilder`, `javax.sql.PooledConnectionBuilder`, `java.sql.ShardingKey`, `java.sql.ShardkingKeyBuilder`
    - ...

java.sql2 (1)

- Egy új aszinkron adatbázis-elérési API.
- Nem a JDBC API kiterjesztése, hanem egy teljesen önálló API.
  - Egyáltalán nem is támaszkodik a JDBC-re.
- Nem egy általános célú adatbázis-elérési API.
  - Nem célja a JDBC helyettesítése.
- Az építő programtervezési minta használata jellemzi.
- Folyékony interfész (*fluent interface*).
  - Martin Fowler. *FluentInterface*. https://martinfowler.com/bliki/FluentInterface.html

java.sql2 (2)

- Lásd:
  - Douglas Surber. *JDBC Next: A New Asynchronous API for Connecting to a Database*. October 3, 2017. https://www.slideshare.net/ypoirier/jdbc-next-a-new-asynchronous-api-for-connecting-to-a-database
  - Package `java.sql2` http://cr.openjdk.java.net/~lancea/apidoc/
  - http://hg.openjdk.java.net/jdk10/sandbox/jdk/file/a31057bda7c5/src/java.sql/share/classes/java/sql2

JDBC és Java típusok megfeleltetése

- Lásd a specifikáció *Data Type Conversion Tables* című B. függelékét.

JDBC meghajtó

- Az a szoftverkomponens, mely lehetővé teszi, hogy egy Java alkalmazás kapcsolatot létesítsen egy adatforrással.
  - A JDBC API egy implementációja egy adatforráshoz.
- A JDBC specifikáció szerint négyféle megvalósítása lehetséges.

JDBC meghajtók fajtái (1)

- **1. típusú**: a JDBC API implementálása egy másik adatforrás elérési API – például az ODBC – használatával történik.
  - Ilyen például a JDBC-ODBC híd, mely a JDBC hívásokat ODBC hívásokká alakítja át.
    - Egy natív programkönyvtárat igényel a kliens oldalon, mely korlátozza a hordozhatóságot.
    - A JDBC-ODBC fordítás miatt romlik a teljesítmény.

JDBC meghajtók fajtái (2)

- **2. típusú**: részben Java-ban implementált meghajtók, melyek az adatforrást egy natív kliens programkönyvtáron keresztül érik el.
  - Egy natív programkönyvtárat igényel a kliens oldalon, mely korlátozza a hordozhatóságot.
  - Jobb teljesítmény (nincs fordítás, mint például a JDBC-ODBC híd esetén).

JDBC meghajtók fajtái (3)

- **3. típusú**: tisztán Java-ban implementált meghajtók, melyek egy köztes rétegen keresztül kommunikálnak az adatforrással egy adatforrás-független protokoll szerint.
  - A köztes réteg fordítja le a kliens kéréseit az adatforrás nyelvére.
  - Nincs szükség a kliens oldalon natív programkönyvtárra.
  - A köztes réteg további hasznos szolgáltatásokat nyújthat (például gyorsítótárazás, naplózás).

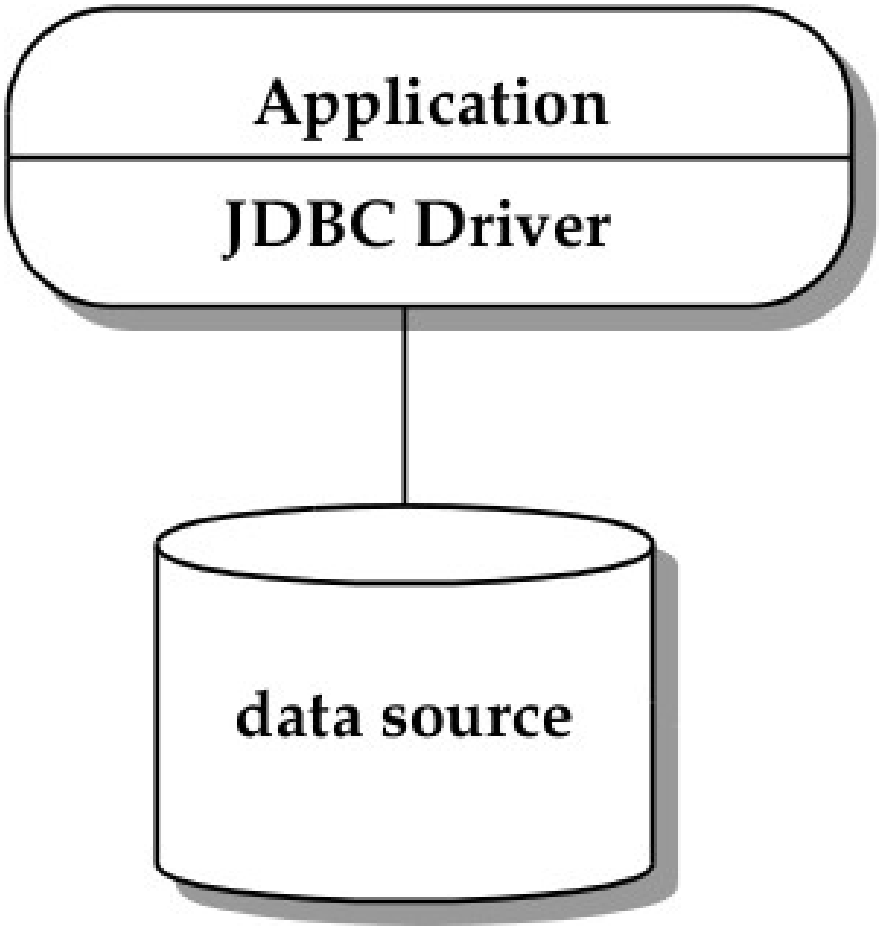
JDBC meghajtók fajtái (4)

- **4. típusú**: tisztán Java-ban implementált JDBC meghajtók, melyek közvetlenül kommunikálnak az adatforrással, például adott hálózati protokoll szerint.
  - Egyik oldalon sem szükséges bármilyen további szoftver.

JDBC meghajtók fajtái (5)

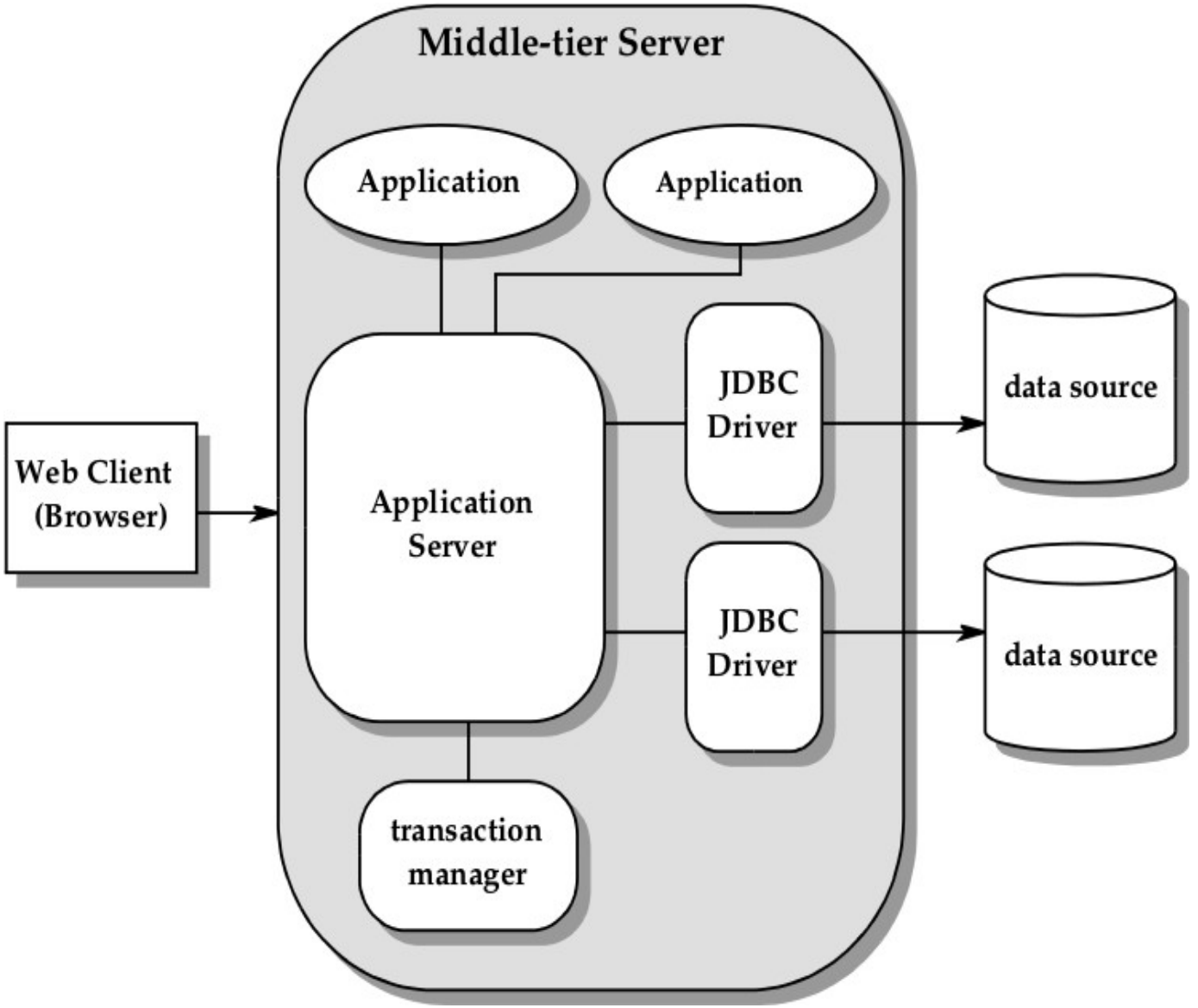
- **2. típusú**:
  - *Oracle Database: JDBC OCI Driver* http://www.oracle.com/technetwork/database/features/instant-client/index-097480.html
  - *SQLite JDBC Driver* https://github.com/xerial/sqlite-jdbc
- **3. típusú**:
  - *Easysoft JDBC-ODBC Bridge* http://www.easysoft.com/products/data_access/jdbc_odbc_bridge/
- **4. típusú**:
  - *Microsoft SQL Server* https://github.com/Microsoft/mssql-jdbc
  - *MySQL: Connector/J* https://dev.mysql.com/downloads/connector/j/
  - *Oracle Database: JDBC Thin Driver* https://www.oracle.com/technetwork/database/application-development/jdbc/downloads/jdbc-ucp-183-5013470.html
  - *PostgreSQL* https://jdbc.postgresql.org/
- **1–4. típusú**:
  - *OpenLink Universal Data Access (UDA) JDBC Drivers* https://uda.openlinksw.com/jdbc/

Kétrétegű modell



Forrás: JDBC 4.3 Specification

Háromrétegű modell



Forrás: JDBC 4.3 Specification

A java.sql.Driver interfész (1)

- JDBC meghajtót reprezentáló interfész.
- Az implementáció kell, hogy tartalmazzon egy statikus inicializáló blokkot, melynek feladata az osztály egy példányának regisztrálása a `java.sql.DriverManager` osztállyal.
  - Ehhez kötelező paraméter nélküli konstruktor.
- Lásd: https://docs.oracle.com/en/java/javase/11/docs/api/java.sql/java/sql/Driver.html

A java.sql.Driver interfész (2)

- Példa az interfész implementálására:

```
public class MyJDBCDriver implements java.sql.Driver {

  static {
    java.sql.DriverManager.registerDriver(new MyJDBCDriver());
  }

  // ...
}
```

Adatbázis kapcsolat létrehozása

- A `java.sql.Connection` interfész ábrázol egy kapcsolatot egy adatforráshoz.
  - Egy alkalmazás egyidejűleg több kapcsolatot is fenntarthat több különböző vagy akár ugyanahhoz az adatforráshoz.
- Lásd: https://docs.oracle.com/en/java/javase/11/docs/api/java.sql/java/sql/Connection.html
- Egy JDBC alkalmazás kétféle módon kapcsolódhat egy adatforráshoz:
  - A `java.sql.DriverManager` osztály révén.
  - A `javax.sql.DataSource` interfész révén (ez az ajánlott).

SQL utasítások végrehajtása

- A `java.sql.Connection` interfész metódusait kell használni.

A java.sql.DriverManager osztály (1)

- Kezeli a kliensek számára rendelkezésre álló JDBC meghajtókat.
- A kliens a `DriverManager` osztály `getConnection()` metódusát használja kapcsolat létrehozásához, melynek egy adatforrást azonosító URL-t ad át paraméterként.
- A `DriverManager` megkeresi a regisztrált JDBC meghajtók közül a megfelelőt, mely létrehozza az adott adatforráshoz a kapcsolatot.
  - Az osztály ehhez sorban végigpróbálja a regisztrált JDBC meghajtókat és meghívja a `connect()` metódusukat, átadva paraméterként az URL-t.
- Lásd: https://docs.oracle.com/en/java/javase/11/docs/api/java.sql/java/sql/DriverManager.html

A java.sql.DriverManager osztály (2)

- Az adatbázis URL formája:
  - `jdbc:`*alprotokoll*`:`*alnév*
  - Példák:
    - `jdbc:hsqldb:mem:testdb`
    - `jdbc:hsqldb:file:testdb`
    - `jdbc:hsqldb:file:/var/db/testdb`
    - `jdbc:hsqldb:hsql://localhost/testdb`
    - `jdbc:hsqldb:http://localhost/testdb`
    - `jdbc:oracle:thin:@db.inf.unideb.hu:1521:ora11g`

A java.sql.DriverManager osztály (3)

- Az osztály az inicializálása során megpróbálja betölteni a `jdbc.drivers` rendszertulajdonság értékeként adott meghajtókat.
- A JDBC 4.0 számú verziója előtt a JDBC meghajtók betöltése az alábbi módon volt elvégezhető az alkalmazásban:

```
Class.forName("pkg.MyJDBCDriver");

    vagy

DriverManager.registerDriver(new pkg.MyJDBCDriver());
```

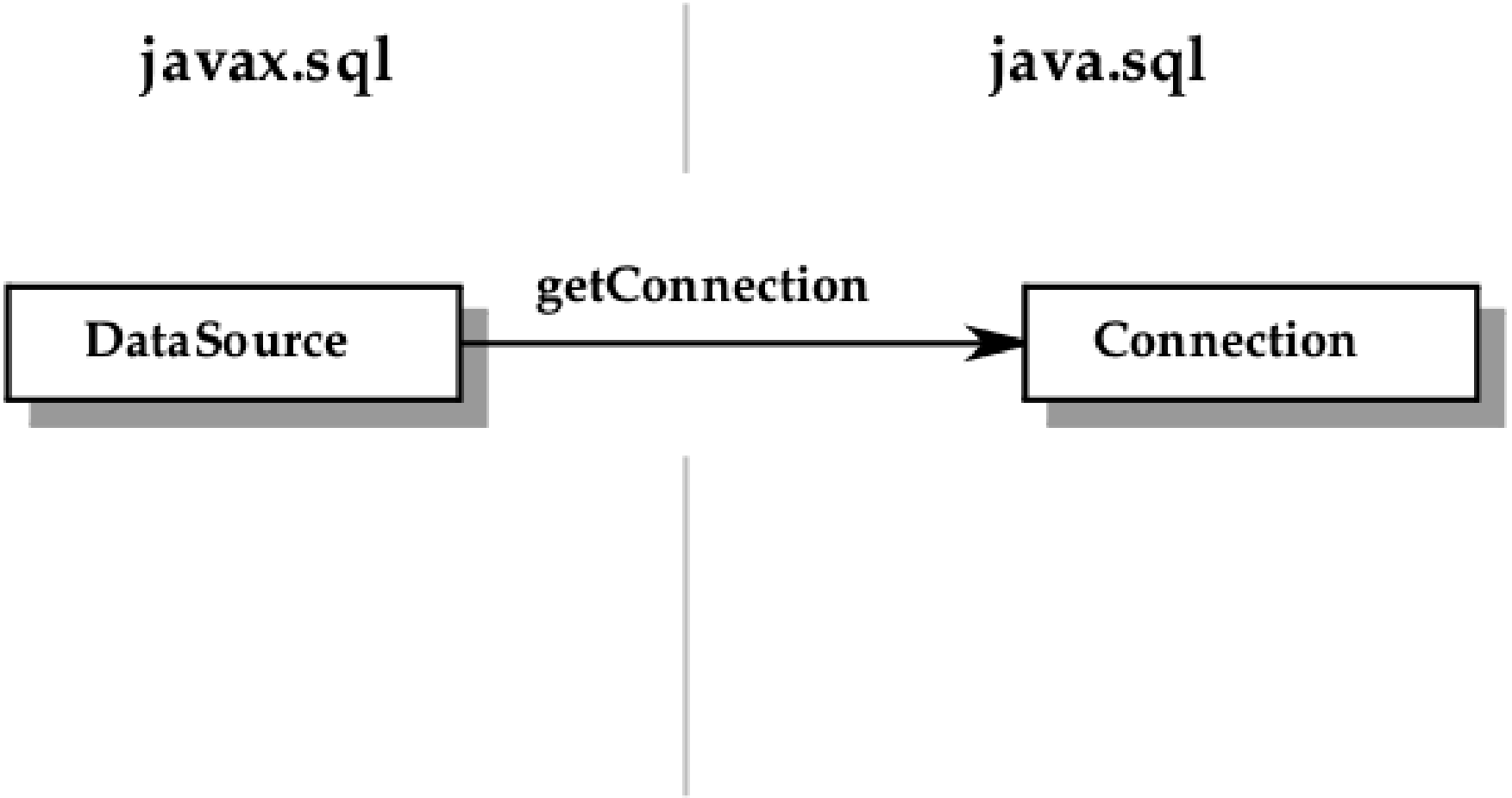
A java.sql.DriverManager osztály (4)

- A JDBC 4.0 számú verziójától nem szükséges a JDBC meghajtó explicit betöltése, a meghajtót tartalmazó JAR állományt egyszerűen hozzá kell adni a *classpath*-hoz.
  - A JAR a `META-INF/services/java.sql.Driver` állományban kell, hogy tartalmazza a `java.sql.Driver` interfészt implementáció osztály teljes minősített nevét.
    - Lásd: *Service Provider* mechanizmus
      - *JAR File Specification* https://docs.oracle.com/en/java/javase/11/docs/specs/jar/jar.html
      - `java.util.ServiceLoader` https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/ServiceLoader.html

A javax.sql.DataSource interfész (1)

- A `java.sql.DriverManager` osztály ajánlott alternatívája kapcsolatok létrehozására.
- Olyan módon növeli az alkalmazások hordozhatóságát, hogy egy logikai nevet használ az adatforrásokhoz.
  - Egy adott logikai nevű objektum egy JNDI névszolgáltatónál regisztrálható és onnan kérhető le.
- Egy fizikai adatforrást ábrázol, melyhez lehetővé teszi kapcsolatok létrehozását.
- Lásd: https://docs.oracle.com/en/java/javase/11/docs/api/java.sql/javax/sql/DataSource.html

A javax.sql.DataSource interfész (2)



Forrás: JDBC 4.3 Specification

A javax.sql.DataSource interfész (3)

- A JDBC az alábbi tulajdonságokat definiálja a `DataSource` implementációkhoz (csak a `description` kötelező):
  - Az implementációk a támogatott tulajdonságokhoz lekérdező és beállító metódusokat kell, hogy biztosítsanak.

Tulajdonság	Típus	Leírás
`databaseName`	`String`	adatbázis neve
`dataSourceName`	`String`	az adatforrás neve
`description`	`String`	az adatforrás leírása
`networkProtocol`	`String`	hálózati protokoll a szerverrel való kommunikációhoz
`password`	`String`	jelszó
`portNumber`	`int`	portszám
`roleName`	`String`	SQL szerepkör
`serverName`	`String`	adatbázis szerver neve
`user`	`String`	felhasználónév

DataSource megvalósítások (1)

- ***Apache Derby/Java DB***: `org.apache.derby.jdbc.ClientDataSource`
  https://db.apache.org/derby/docs/10.14/publishedapi/org/apache/derby/jdbc/ClientDataSource.html
- ***HSQLDB***: `org.hsqldb.jdbc.JDBCDataSource`
  http://hsqldb.org/doc/src/org/hsqldb/jdbc/JDBCDataSource.html
- ***Microsoft SQLServer***:
  `com.microsoft.sqlserver.jdbc.SQLServerDataSource`
  http://javadoc.io/doc/com.microsoft.sqlserver/mssql-jdbc/7.1.3.jre11-preview
- ***Oracle Database***: `oracle.jdbc.pool.OracleDataSource`
  https://docs.oracle.com/en/database/oracle/oracle-database/18/jajdb/oracle/jdbc/pool/OracleDataSource.html
- ***PostgreSQL***: `org.postgresql.ds.PGSimpleDataSource`
  https://jdbc.postgresql.org/documentation/publicapi/org/postgresql/ds/PGSimpleDataSource.html
- …

DataSource megvalósítások (2)

- Példa `DataSource` objektum létrehozására (Java DB):

```
import org.apache.derby.jdbc.ClientDataSource;

public javax.sql.DataSource getDataSource() {
  ClientDataSource cds = new ClientDataSource();
  cds.setDatabaseName("testDB");
  cds.setDescription("Test database");
  cds.setServerName("localhost");
  cds.setPortNumber(1527);
  cds.setUser("guest");
  cds.setPassword("secret");
  cds.setSecurityMechanism(
    ClientDataSource.STRONG_PASSWORD_SUBSTITUTE_SECURITY);
  return cds;
}
```

DataSource használata (1)

- *Java Naming and Directory Interface* (JNDI) http://www.oracle.com/technetwork/java/jndi/
  - *JNDI Interface-related APIs and Developer Guides* http://docs.oracle.com/javase/8/docs/technotes/guides/jndi/
  - *The Java Tutorials: Trail: Java Naming and Directory Interface* https://docs.oracle.com/javase/tutorial/jndi/
  - A `javax.naming` csomag és alcsomagjai tartalmazzák. https://docs.oracle.com/en/java/javase/11/docs/api/java.naming/javax/naming/package-summary.html

DataSource használata (2)

- `DataSource` objektum regisztrálása JNDI névszolgáltatónál:

```
VendorDataSource vds = new VendorDataSource();
vds.setServerName("localhost");
vds.setDatabaseName("testDB");
vds.setDescription("Data source for testing");

Context ctx = new InitialContext();
ctx.bind("jdbc/testDB", vds);
```

DataSource használata (3)

- Kapcsolat létrehozása JNDI névszolgáltatónál regisztrált `DataSource` objektummal:

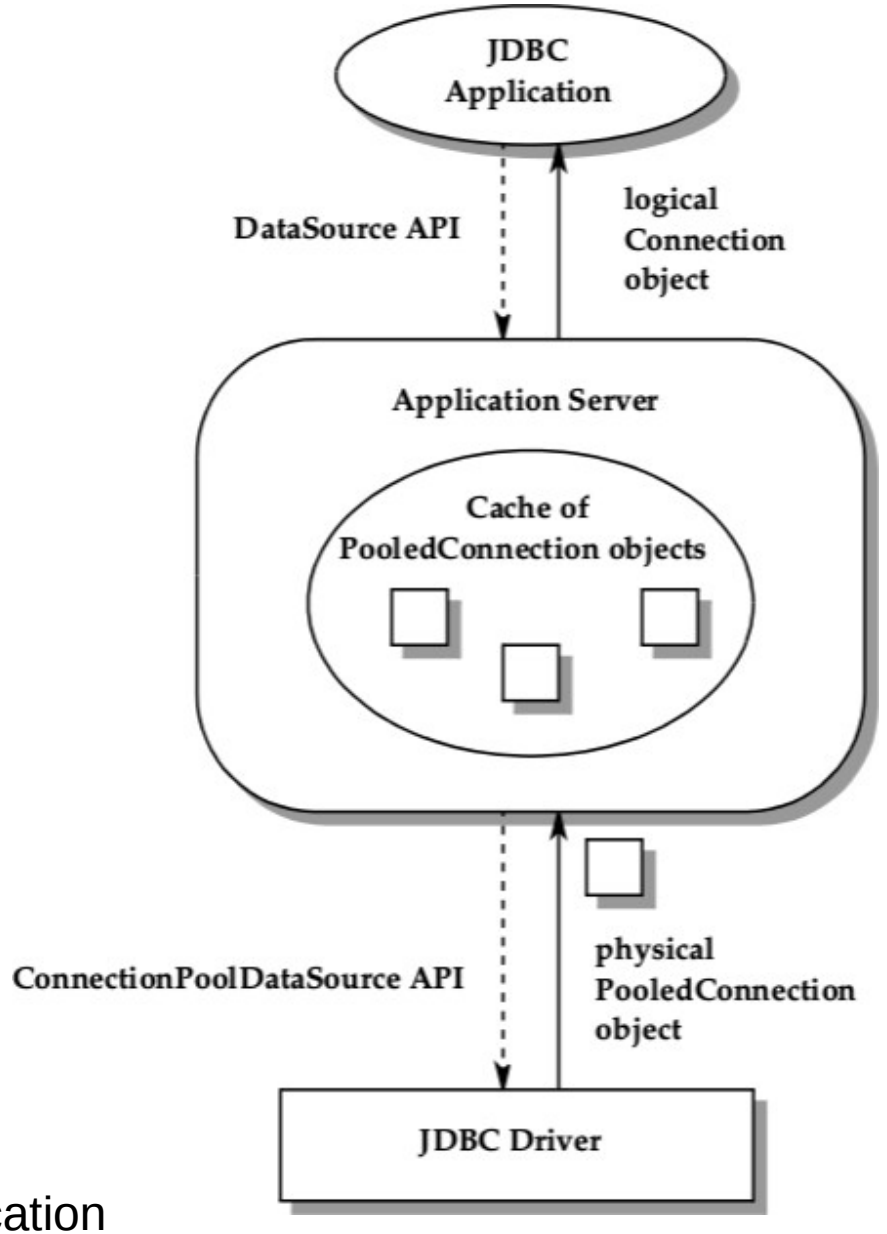
```
Context ctx = new InitialContext();

DataSource ds = (DataSource) ctx.lookup("jdbc/testDB");
Connection conn = ds.getConnection("jeszy", "secret");
```

Connection Pooling (1)

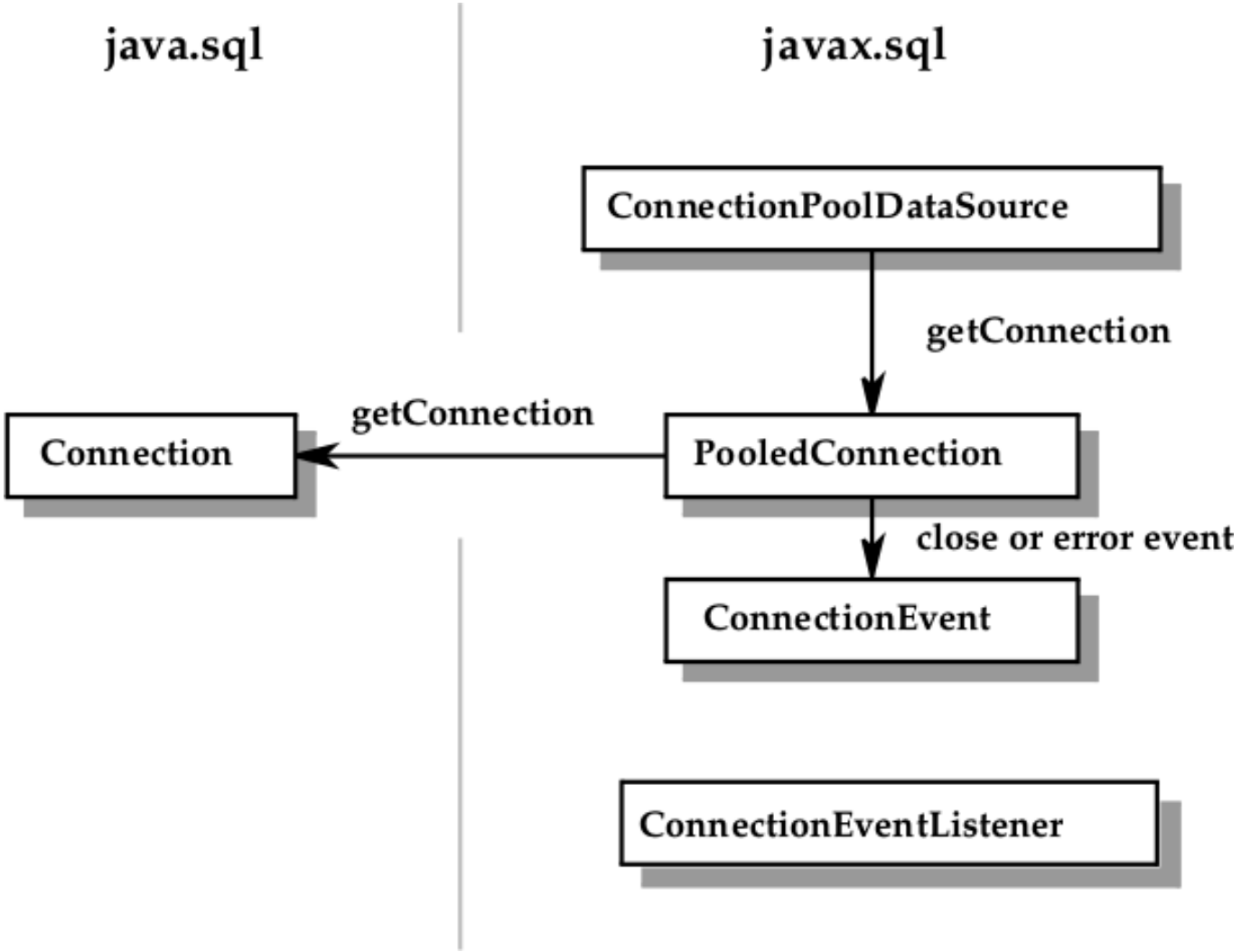
- Egy egyszerű `DataSource` implementációnál 1:1 megfeleltetés van a kliens `Connection` objektuma és a fizikai adatbázis kapcsolat között.
  - A kapcsolat lezárásakor megszűnik a fizikai kapcsolat.
- Egy *connection pool* fizikai adatbázis kapcsolatokat gyorsítótáraz, melyek több kliens munkameneten keresztül újrafelhasználhatóak.
  - Ez a teljesítményt és a skálázhatóságot növeli, különösen a háromrétegű környezetekben.
- A *connection pool* megvalósítása implementáció specifikus, a kliensek számára olyan `DataSource` implementáció biztosítása, mely a kapcsolatok gyorsítótárazását transzparenssé teszi.

Connection Pooling (2)



Forrás: JDBC 4.3 Specification

Connection Pooling (3)



Forrás: JDBC 4.3 Specification

Connection Pooling (4)

- A `javax.sql.ConnectionPoolDataSource` interfész:
  - `javax.sql.PooledConnection` objektumok létrehozására szolgál.
- Lásd: https://docs.oracle.com/en/java/javase/11/docs/api/java.sql/javax/sql/ConnectionPoolDataSource.html

Connection Pooling (5)

- A `javax.sql.PooledConnection` interfész:
  - Egy fizikai kapcsolatot ábrázol egy adatforráshoz, mely újrafelhasználható.
    - A fizikai kapcsolat nem kerül lezárásra, amikor egy kliensnek nincs már rá szüksége, hanem visszakerül a *connection pool*-ba.
  - Az alkalmazásfejlesztők nem használják közvetlenül, az alkalmazásszerver kezeli.
    - Az alkalmazásszerver a saját `DataSource` implementációja mögé rejti.
      - Amikor a kliens meghívja a `DataSource` objektum `getConnection()` metódusát, a `PooledConnection` objektumot használja az alkalmazásszerver egy logikai kapcsolat létrehozásához.
    - A logikai kapcsolat lezárásakor a fizikai kapcsolat nem lesz lezárva.
  - Lásd: https://docs.oracle.com/en/java/javase/11/docs/api/java.sql/javax/sql/PooledConnection.html

Connection Pooling (6)

- A *connection pool* kezelője úgy értesül egy `PooledConnection` objektumokkal. kapcsolatos eseményekről, hogy implementálja a `javax.sql.ConnectionEventListener` interfészt és a `PooledConnection` objektum figyelőjeként regisztrálja magát.
  - A regisztrálás a `PooledConnection` interfész `addConnectionEventListener()` metódusával történik.

Connection Pooling (7)

- Utasítások gyorsítótárazása:
  - A kapcsolatokhoz hasonlóan `java.sql.PreparedStatement` objektumok is gyorsítótárazhatóak.
  - Ez is a kliens számára transzparens módon történik.

Connection Pooling (8)

- A JDBC az alábbi tulajdonságokat definiálja a `ConnectionPoolDataSource` implementációkhoz:
  - Az implementációk a támogatott tulajdonságokhoz lekérdező és beállító metódusokat kell, hogy biztosítsanak.

Tulajdonság	Típus	Leírás
`maxStatements`	`int`	a nyitva tartandó utasítások száma
`initialPoolSize`	`int`	a fizikai kapcsolatok száma a *connection pool* létrehozásakor
`minPoolSize`	`int`	a fizikai kapcsolatok minimális száma
`maxPoolSize`	`int`	a fizikai kapcsolatok maximális száma
`maxIdleTime`	`int`	mennyi idő után kell lezárni egy nem használt fizikai kapcsolatot (másodperc)
`propertyCycle`	`int`	mennyi időnek kell eltelnie a fenti tulajdonságok által meghatározott „házirend” kikényszerítése előtt (másodperc)

Connection Pooling (9)

- Megvalósítások:
  - *Apache Commons DBCP* (licenc: *Apache License 2.0*) https://commons.apache.org/proper/commons-dbcp/
  - *C3P0* (licenc: LGPLv2.1) https://www.mchange.com/projects/c3p0/
  - *HikariCP* (licenc: *Apache License 2.0*) https://brettwooldridge.github.io/HikariCP/
  - *Apache Tomcat* (licenc: *Apache License 2.0*):
    - *Tomcat JDBC Connection Pool* http://tomcat.apache.org/tomcat-8.5-doc/jdbc-pool.html
  - *Universal Connection Pool* (UCP) https://www.oracle.com/technetwork/database/application-development/jdbc/downloads/index.html
  - *Vibur DBCP* (licenc: *Apache License 2.0*) http://www.vibur.org/

Connection pool használata (1)

- A JNDI névszolgáltatónál egy olyan `DataSource` objektumot kell regisztrálni, melyen keresztül egy kliens a *connection pool*-tól kérhet logikai kapcsolatokat:

```
import com.zaxxer.hikari.HikariDataSource;

HikariDataSource ds = new HikariDataSource();
ds.setJdbcUrl("jdbc:hsqldb:hsql://localhost/testDB");
ds.setUsername("guest");
ds.setPassword("secret");
ds.setMaximumPoolSize(10);

Context ctx = new InitialContext();
ctx.bind("jdbc/testDB", ds);
```

Connection pool használata (2)

- Kapcsolat létrehozása JNDI névszolgáltatónál regisztrált `DataSource` objektummal:

```
Context ctx = new InitialContext();
DataSource ds = (DataSource) ctx.lookup("jdbc/testDB");
Connection conn = ds.getConnection();
```

Adatforrás konfigurálás (1)

- *Apache Tomcat 8.5*:
  - *JNDI Datasource HOW-TO*
    http://tomcat.apache.org/tomcat-8.5-doc/jndi-datasource-examples-howto.html

Adatforrás konfigurálás (2)

```
<!--/META-INF/context.xml -->
<Context>
  <Resource name="jdbc/PgDS"
    auth="Container"
    type="javax.sql.DataSource"
    driverClassName="org.postgresql.Driver"
    url="jdbc:postgresql://localhost:5432/testdb"
    username="guest"
    password="secret"
    maxTotal="20"
    maxIdle="10"
    maxWaitMillis="-1"/>
</Context>
```

```
<!-- /WEB-INF/web.xml -->
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee" version="3.1">
  <resource-ref>
    <description>PostgreSQL test database</description>
      <res-ref-name>jdbc/PgDS</res-ref-name>
      <res-type>javax.sql.DataSource</res-type>
      <res-auth>Container</res-auth>
    </resource-ref>
</web-app>
```

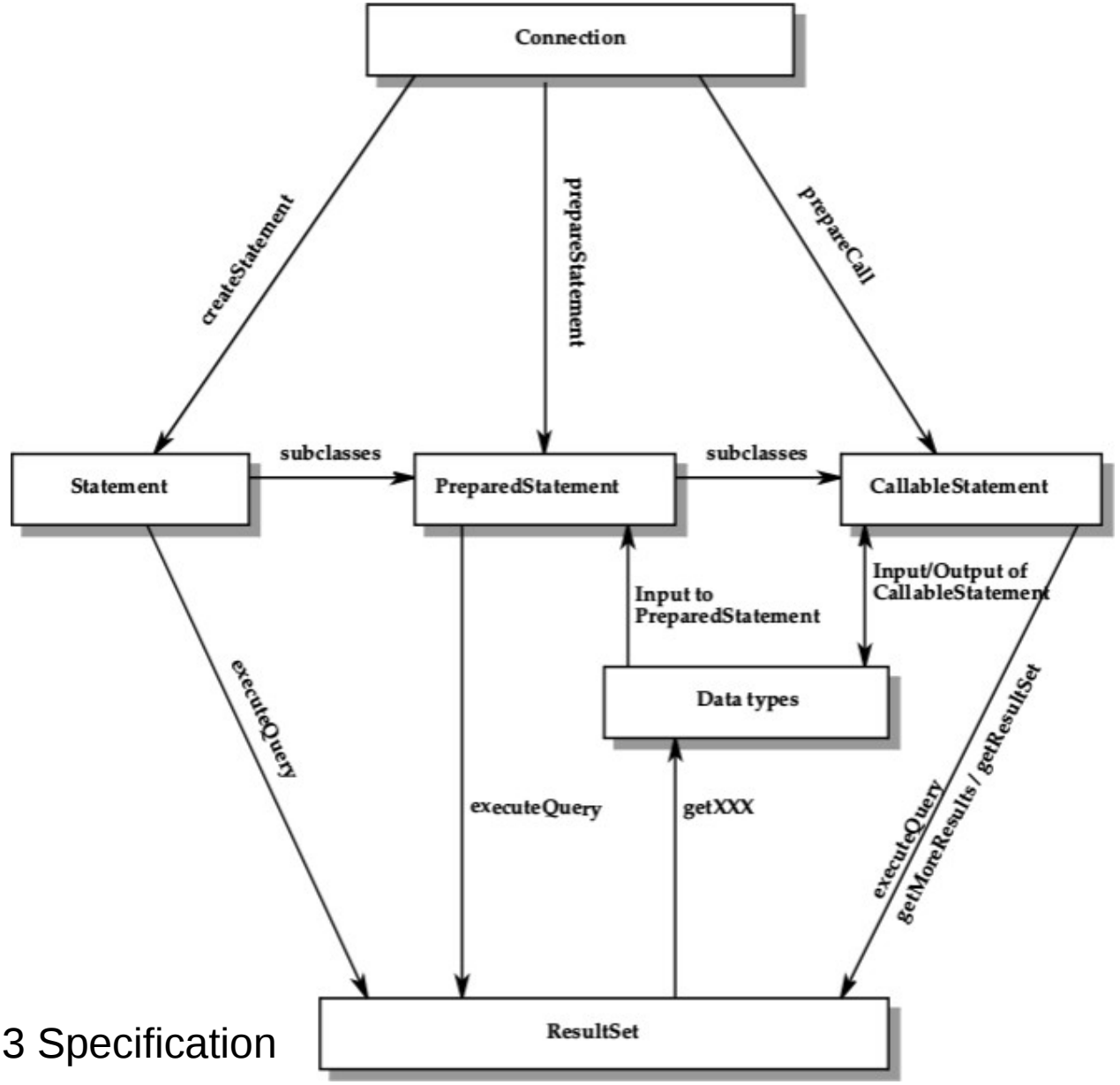
Adatforrás konfigurálás (3)

- *WildFly 15*:
  - *WildFly Admin Guide – Subsystem configuration – DataSource Configuration* http://docs.wildfly.org/15/Admin_Guide.html#DataSource

Adatforrás konfigurálás (4)

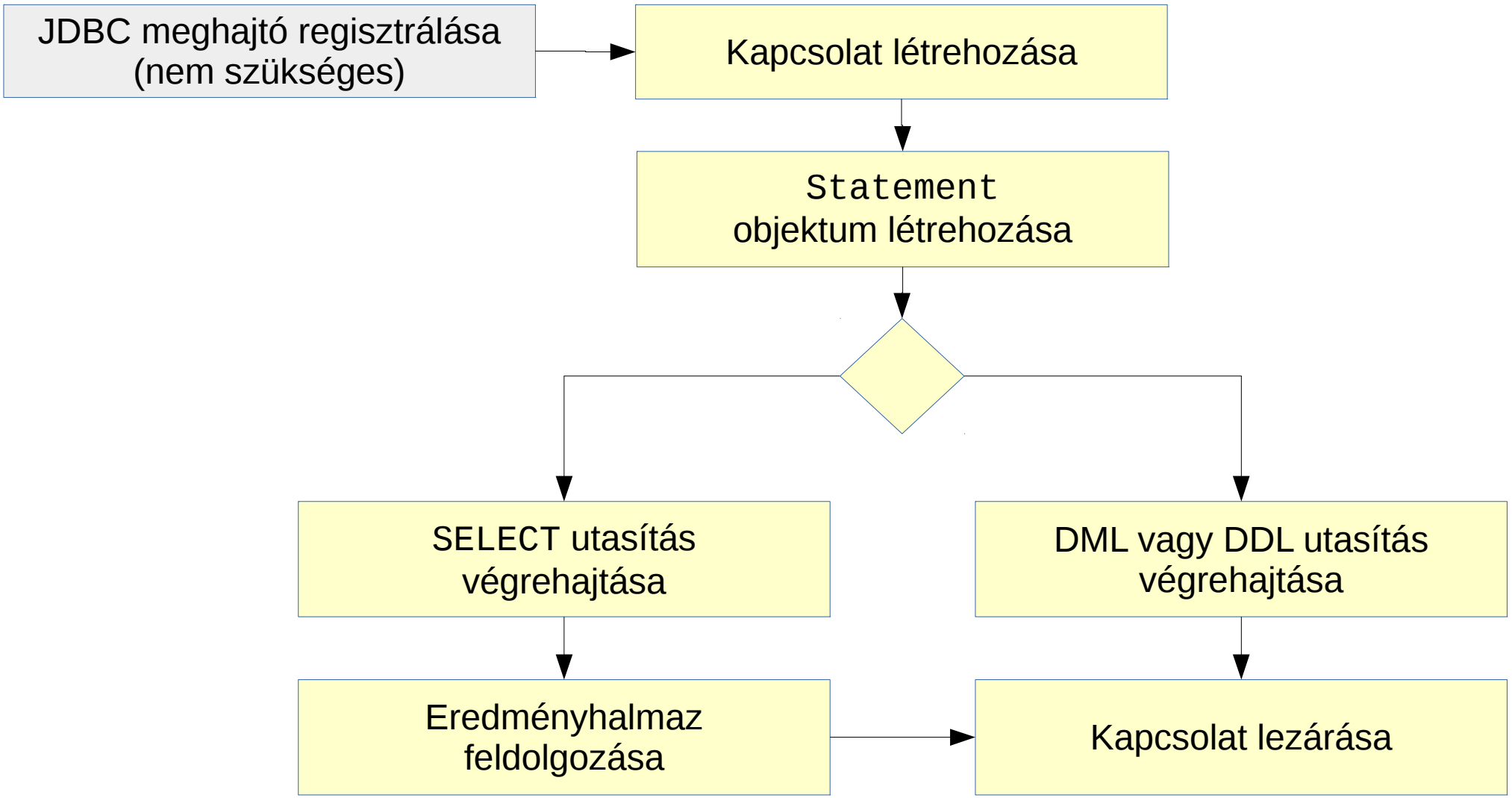
```
<subsystem xmlns="urn:jboss:domain:datasources:4.0">
  <datasources>
    <datasource jndi-name="java:datasources/PgDS" pool-name="PgDS">
      <connection-
url>jdbc:postgresql://localhost:5432/testdb</connection-url>
      <driver>postgresql</driver>
      <pool>
        <min-pool-size>10</min-pool-size>
        <max-pool-size>20</max-pool-size>
        <prefill>true</prefill>
      </pool>
      <security>
        <user-name>guest</user-name>
        <password>secret</password>
      </security>
    </datasource>
    <drivers>
      <driver name="postgresql" module="org.postgresql">
        <driver-class>org.postgresql.Driver</driver-class>
        <datasource-
class>org.postgresql.ds.PGSimpleDataSource</datasource-class>
      </driver>
    </drivers>
  </datasources>
</subsystem>
```

A java.sql csomag



Forrás: JDBC 4.3 Specification

SQL utasítások végrehajtásának lépései



Statement (1)

- Statikus (nem paraméterezhető) SQL utasítást reprezentáló interfész.
- A `java.sql.Connection` interfész `createStatement()` metódusával hozható létre egy `Statement` objektum.
- Ha már nincs szükség rá, zárjuk le a `close()` metódussal.
  - Ezután a `close()` és az `isClosed()` metódusok kivételével a `Statement` objektum minden metódusának meghívása `SQLException` kivételt eredményez.
  - A kapcsolat lezárásakor automatikusan lezárásra kerül valamennyi általa létrehozott `Statement` objektum.
- Lásd: https://docs.oracle.com/en/java/javase/11/docs/api/java.sql/java/sql/Statement.html

Statement (2)

- Egyetlen eredményhalmazt visszaadó SQL utasítás végrehajtása:
  - `java.sql.ResultSet executeQuery(String sql);`
- DML vagy DDL utasítás végrehajtása:
  - `int executeUpdate(String sql);`
    `long executeLargeUpdate(String sql);`
    - A visszatérési érték DML utasításoknál a módosított sorok száma, DDL utasításoknál 0.
- Tetszőleges SQL utasítás végrehajtása:
  - `boolean execute(String sql);`
    - A visszatérési érték `true`, ha az első eredmény egy eredményhalmaz, egyébként `false`.
    - Előbbi esetben a `getResultSet()` metódus szolgál az eredmény lekérdezésére, utóbbi esetben a `getUpdateCount()` és a `getLargeUpdateCount()` metódus.

Statement (3)

- Egyetlen eredményhalmazt visszaadó SQL utasítás végrehajtása:

```
Connection conn;

Statement st = conn.createStatement();
ResultSet rs = st.executeQuery("SELECT username, last_login
  FROM users ORDER BY last_login");
// ...

st.close();
```

Statement (4)

- DML vagy DDL utasítás végrehajtása:

```
Connection conn;

Statement st = conn.createStatement();
int rows = st.executeUpdate("UPDATE users SET password = 'secret'
  WHERE username = 'jeszy'");
System.out.printf("Updated %d row(s)\n", rows);
st.close();
```

```
Connection conn;

Statement st = conn.createStatement();
int rows = st.executeUpdate("DELETE FROM users WHERE last_login IS NULL");
System.out.printf("Deleted %d row(s)\n", rows);
st.close();
```

Statement (5)

- Tetszőleges SQL utasítás végrehajtása:

```
Connection conn;
String sql;

Statement st = conn.createStatement();
boolean isQuery = st.execute(sql);
if (isQuery) {
  // lekérdezés
  ResultSet rs = st.getResultSet();
  // ...
} else {
  // DML vagy DDL utasítás
  int updateCount = st.getUpdateCount();
  // ...
}
st.close();
```

ResultSet (1)

- Egy lekérdezés eredményhalmazát reprezentálja.
- Egy kurzorral mutatja az aktuális sort, mely kezdetben az első sor előtt áll.
  - A `next()` metódus mozgatja a kurzort a következő sorra.
    - Visszatérési értéke `true`, ha van következő sor, egyébként `false`.
- Alapértelmezésben csak előre navigálható, nem módosítható.
- Lásd: https://docs.oracle.com/en/java/javase/11/docs/api/java.sql/java/sql/ResultSet.html

ResultSet (2)

- Az aktuális sor oszlopértékeinek lekérdezése:
  - A `get`*`XXX`*`(`*`index`*`)` és a `get`*`XXX`*`(`*`név`*`)` metódusok szolgálnak erre a célra, ahol *XXX* egy típus neve.
    - Például: `getBigDecimal()`, `getInt()`, `getLong()`, `getString()`, `getObject()`, ...
    - Ha a visszatérési érték primitív típusú, akkor a `wasNull()` metódussal állapítható meg, hogy `NULL`-e az utoljára lekérdezett oszlopérték.
    - Ha a visszatérési érték egy objektum, akkor `null` referencia jelzi a `NULL` értéket.
  - Az első oszlop indexe 1.

ResultSet (3)

- Ha már nincs szükség rá, akkor a `close()` metódussal zárjuk le.
  - Ezután a `close()` és `isClosed()` metódusok kivételével a `ResultSet` objektum minden metódusának meghívása `SQLException` kivételt eredményez.
- Egy `Statement` objektum vagy a kapcsolat lezárásakor automatikusan lezárásra kerül valamennyi általa létrehozott `ResultSet` objektum.
  - A `ResultSet` objektumok lezárását eredményezi, ha a létrehozó `Statement` objektummal ismét végrehajtunk egy utasítást.

ResultSet (4)

- Példa `ResultSet` feldolgozására:

```
Connection conn;

Statement st = conn.createStatement();
ResultSet rs = st.executeQuery("select username, last_login FROM users
  WHERE CAST(last_login AS DATE) = CURRENT_DATE ORDER BY last_login");
while (rs.next()) {
  System.out.println("Username: " + rs.getString(1));
  System.out.println("Last login: " + rs.getTimestamp(2));
}
rs.close();
st.close();
```

PreparedStatement (1)

- A `java.sql.Statement` interfészt terjeszti ki, egy előfordított SQL utasítást ábrázol.
- Akkor használjuk, ha egy utasítást többször is végre szeretnénk hajtani.
- Paraméterezhető:
  - Az utasításban `'?'` karakterek jelölik a paramétereket.
  - A végrehajtás előtt kell megadni a paraméterek értékét.
- Lásd: https://docs.oracle.com/en/java/javase/11/docs/api/java.sql/java/sql/PreparedStatement.html

PreparedStatement (2)

- Paraméterek értékének beállítása:
  - A `set`*`XXX`*`(`*`index`*`,`*`érték`*`)` metódusok szolgálnak erre a célra, ahol *XXX* egy típus neve.
    - Például: `setBigDecimal()`, `setInt()`, `setLong()`, `setString()`, `setObject()`, ...
    - Az első paraméter indexe 1.
  - `NULL` beállítása paraméter értékeként a `setNull(`*`index`*`,`*`típus`*`)` metódussal.
    - Például:
      - `ps.setNull(1, java.sql.Types.VARCHAR);`

PreparedStatement (3)

```
Connection conn;

PreparedStatement ps = conn.prepareStatement("UPDATE users
  SET last_login = ? WHERE username = ?");
ps.setTimestamp(1, Timestamp.valueOf(LocalDateTime.now()));
ps.setString(2, "jeszy");
int rows = ps.executeUpdate();
System.out.printf("Updated %d row(s)\n", rows);
ps.close();
```

```
Connection conn;

PreparedStatement ps = conn.prepareStatement("SELECT *
  FROM users WHERE username = ?");
ps.setString(1, "jeszy");
ResultSet rs = ps.executeQuery();
// ...

rs.close();
ps.close();
```

Escape-szintaxis (1)

- SQL utasításokban használható a `java.sql.Statement` interfésznél és leszármazottjainál.
  - Olyan elterjedten használt lehetőségekhez, melyeknek nincs szabványos SQL szintaxisa vagy melyek szintaxisa gyártófüggő.
  - Az adatforrás által nem támogatott, de a JDBC meghajtó által implementált lehetőségekhez.

Escape-szintaxis (2)

- **Skalár függvények**: `{fn` *`név`*`(`*`args`*`)}`
  - A specifikáció C. függeléke tartalmazza azon skalár függvények listáját, melyeket a JDBC meghajtók várhatóan támogatnak.
    - Példa: `{fn concat('Hello', ' world!')}`, `{fn now()}`, `{fn user()}`
- **Dátum és idő literálok**
  - Példa: `{d '1848-03-15'}`, `{t '12:00:00'}`, `{ts '2014-01-14 10:00:00'}`
- **Külső összekapcsolások**
- **Tárolt eljárások és függvények meghívása**: `{call` *`név[`*`(`*`args`*`)]`*`}` vagy `{?= call` *`név[`*`(`*`args`*`)]`*`}`
- **Escape karakter megadása `LIKE` klózhoz**: `{escape '`*`karakter`*`'}`
  - Példa: `"SELECT * FROM users WHERE password LIKE '%\\_%' {escape '\\'}"`

Kivételkezelés (1)

```
Connection conn = null;
try {
  conn = DriverManager.getConnection(url, user, password);
  Statement st = conn.createStatement();
  ResultSet rs = st.executeQuery(query);
  while(rs.next()){
    // ...
  }
  rs.close();
  st.close();
  conn.close();
} catch(SQLException e) {
  e.printStackTrace();
} finally {
  try {
    if(conn != null) conn.close();
  } catch(SQLException e){
    e.printStackTrace();
  }
}
```

Kivételkezelés (2)

- A `Connection`, `ResultSet` és `Statement` interfészek implementálják a `java.lang.AutoCloseable` interfészt, így használhatóak a Java SE 7-ben megjelent *try-with-resources* utasításban:

```
try (Connection conn = DriverManager.getConnection(url,
    user, password)) {

  // ...

} catch(SQLException e) {
  e.printStackTrace();
}
```

Kivételek (1)

- A `java.sql.SQLException` osztály és alosztályai szolgáltatnak információkat az adatforrás elérése során fellépő hibákról és figyelmeztetésekről.
  - Ha több hiba történik, akkor ezek láncolása, a `getNextException()` metódus szolgáltatja a lánc következő elemét.
- Lásd: https://docs.oracle.com/en/java/javase/11/docs/api/java.sql/java/sql/SQLException.html

Kivételek (2)

- A hibák és okaik láncának feldolgozása:

```
try {
  // ...
} catch(SQLException e) {
  while(e != null) {
    System.out.println("SQLState: " + e.getSQLState());
    System.out.println("Error Code: " + e.getErrorCode());
    System.out.println("Message: " + e.getMessage());
    Throwable t = e.getCause();
    while(t != null) {
      System.out.println("Cause: " + t);
      t = t.getCause();
    }
    e = e.getNextException();
  }
}
```

Kivételek (3)

- A hibák és okaik láncának feldolgozása:

```
try {
  // ...
} catch(SQLException e) {
  for(Throwable t : e) {
    System.out.println("Error encountered: " + e);
  }
}
```

Kivételek (4)

- A `java.sql.SQLWarning` osztály a `java.sql.SQLException` alosztálya, mely figyelmeztetéseket ábrázol.
  - A `java.sql.Connection`, `java.sql.Statement` és `java.sql.ResultSet` interfészek metódusai eredményezhetnek figyelmeztetéseket, melyek a `getWarnings()` metódussal kérdezhetők le.
    - A figyelmeztetések a hibákhoz hasonlóan történő láncolása.
- Lásd: https://docs.oracle.com/en/java/javase/11/docs/api/java.sql/java/sql/SQLWarning.html

Kivételek (5)

- A `java.sql.BatchUpdateException` osztály a `java.sql.SQLException` alosztálya, mely utasítások kötegelt végrehajtása során bekövetkező hibákról szolgáltat információkat.
- Lásd: https://docs.oracle.com/en/java/javase/11/docs/api/java.sql/java/sql/BatchUpdateException.html

Tranzakció-kezelés (1)

- Az alábbi lehetőségeket biztosítja a JDBC API:
  - *Auto-commit* mód
  - Tranzakció-izolációs szintek
  - Mentési pontok

Tranzakció-kezelés (2)

- A JDBC meghajtó vagy az adatforrás dönti el, hogy mikor kell új tranzakciót kezdeni.
  - Akkor kell egy új tranzakciót kezdeni, amikor az aktuális SQL utasítás számára szükség van egyre és nincs folyamatban lévő tranzakció.
- A kapcsolatok *auto-commit* tulajdonsága határozza meg, hogy mikor van vége egy tranzakciónak.
  - Az *auto-commit* bekapcsolása esetén a tranzakció véglegesítését eredményezi minden egyes SQL utasítás végrehajtásának befejezése.
    - Például egy SELECT utasítás végrehajtása akkor fejeződik be, amikor a hozzá tartozó eredménytábla lezárásra kerül.

Tranzakció-kezelés (3)

- Az *auto-commit* alapértelmezésben be van kapcsolva, amikor egy kapcsolat létrejön.
- Az *auto-commit* kikapcsolása:
  - `Connection conn;`
    `conn.setAutoCommit(false);`
    - Ezt követően minden egyes tranzakciót véglegesíteni kell a `java.sql.Connection` interfész `commit()` metódusával vagy pedig vissza kell görgetni az interfész `rollback()` metódusával.

JDBC a Java EE platformban

- Amikor Java EE komponensek – például JSP vagy EJB komponensek – használják a JDBC API-t, akkor a konténer kezeli a tranzakcióikat és adatforrásaikat.
  - Ilyenkor a fejlesztő nem használja közvetlenül a JDBC API tranzakció és adatforrás kezelési lehetőségeit.

Adatbázis metaadatok (1)

- A `java.sql.DatabaseMetaData` interfész szolgál az adatforrás jellemzőinek lekérdezésére.
  - 170-nél több metódusa van!
- Lásd: https://docs.oracle.com/en/java/javase/11/docs/api/java.sql/java/sql/DatabaseMetaData.html

Adatbázis metaadatok (2)

- **Általános információk lekérdezése**:
  - Például `getDatabaseProductName()`, `getDatabaseProductVersion()`, `getDriverMajorVersion()`, `getDriverMinorVersion()`, ...
- **Lehetőségek támogatottságának lekérdezése**: `supports`*`XXX`*`()` metódusok
  - Például `supportsGroupBy()`, `supportsOuterJoins()`, `supportsStoredProcedures()`, ...
- **Adatforrás korlátok lekérdezése**: `getMax`*`XXX`*`()` metódusok
  - Például `getMaxRowSize()`, `getMaxColumnNameLength()`, `getMaxConnections()`, ...
- **SQL objektumok és attribútumaik lekérdezése**:
  - Például `getSchemas()`, `getTables()`, `getPrimaryKeys()`, `getProcedures()`, ...
- **Tranzakció-kezelés jellemzőinek lekérdezése**:
  - Például `getDefaultTransactionIsolation()`, `supportsMultipleTransactions()`

Adatbázis metaadatok (3)

- Példa az interfész használatára:

```
Connection conn;
DatabaseMetaData  dmd = conn.getMetaData();

String timeDateFunctions = dmd.getTimeDateFunctions();

int jdbcMajorVersion = dmd.getJDBCMajorVersion();
int jdbcMinorVersion = dmd.getJDBCMinorVersion()

boolean storedProcedures = dmd.supportsStoredProcedures();
```

Java 8 dátum és idő típusok

- Nem használhatók közvetlenül a `java.sql.PreparedStatement` és a `java.sql.ResultSet` interfészekkel, helyettük a `java.sql.Date` és a `java.sql.Timestamp` osztályokat kell használni.
- Konverzió szükségeses:
  - **`java.sql.Date`**: lásd a `toLocalDate()` és `valueOf(LocalDate date)` metódusokat.
  - **`java.sql.Timestamp`**: lásd a `toInstant()`, `from(Instant instant)`, `toLocalDateTime()` és `valueOf(LocalDateTime dateTime)` metódusokat.

Adatbázis-kezelő rendszerek népszerűsége

- *DB-Engines Ranking*
  https://db-engines.com/en/ranking
  - *DB-Engines Ranking of Relational DBMS*
    https://db-engines.com/en/ranking/relational+dbms
  - *Popularity of open source DBMS versus commercial DBMS*
    https://db-engines.com/en/ranking_osvsc

Relációs adatbázis-kezelő rendszerek

- Nem szabad:
  - *Microsoft SQL Server*
  - *MySQL Standard Edition, MySQL Enterprise Edition, MySQL Cluster CGE*
  - *Oracle Database*
  - …
- Szabad és nyílt forrású:
  - *MySQL Community Edition*
  - *PostgreSQL*
  - *SQLite*
  - *H2*
  - *HSQLDB*
  - *Apache Derby/Java DB*
  - …

Nem szabad relációs adatbázis-kezelő rendszerek (1)

- *Microsoft SQL Server*
  http://www.microsoft.com/sqlserver/
  - Fejlesztő: Microsoft
  - Platform: Windows, Linux
  - Programozási nyelv: C, C++
  - Kiadások: *Express*, *Developer*, *Standard*, *Enterprise*
    - *Express*: korlátozott lehetőségekkel rendelkező szabadon használható kiadás, például legfeljebb 10 GB méretű adatbázis.
    - *Developer*: a vállalati kiadás valamennyi funkcióját tartalmazza, azonban kizárólag fejlesztéshez és teszteléshez használható.
    - Lásd: https://www.microsoft.com/en-us/sql-server/sql-server-2017-editions

Nem szabad relációs adatbázis-kezelő rendszerek (2)

- *MySQL* https://www.mysql.com/
  - Fejlesztő: *Oracle Corporation*
  - Platform: FreeBSD, Linux, macOS, Solaris, Windows
  - Programozási nyelv: C, C++
  - Kiadások:
    - *MySQL Standard Edition* https://www.mysql.com/products/standard/
    - *MySQL Enterprise Edition* https://www.mysql.com/products/enterprise/ https://www.oracle.com/mysql/enterprise/index.html
    - *MySQL Cluster CGE* https://www.mysql.com/products/cluster/
    - *MySQL Community Edition* https://www.mysql.com/products/community/
  - Kiadások összehasonlítása: https://www.mysql.com/products/

Nem szabad relációs adatbázis-kezelő rendszerek (3)

- *Oracle Database* https://www.oracle.com/database/
  - Fejlesztő: *Oracle Corporation*
  - Platform: Unix-szerű, Windows
  - Programozási nyelv: Assembly, C, C++
  - Kiadások:
    - *Oracle Database 18c Express Edition (XE)*
      - Korlátozott lehetőségekkel rendelkező szabadon használható kiadás.
        - Legfeljebb 12 GB felhasználói adat tárolása, 2 GB RAM és 2 CPU használata.

Szabad és nyílt forrású relációs adatbázis-kezelő rendszerek (1)

- *MySQL Community Server*
  https://dev.mysql.com/downloads/mysql/
  - Fejlesztő: *Oracle Corporation*
  - Platform: FreeBSD, Linux, macOS, Solaris, Windows
  - Programozási nyelv: C, C++
  - Licenc: GPLv2

Szabad és nyílt forrású relációs adatbázis-kezelő rendszerek (2)

- *PostgreSQL* https://www.postgresql.org/
  - Fejlesztő: *PostgreSQL Global Development Group*
  - Platform: Unix-szerű, Windows
  - Programozási nyelv: C
  - Licenc: *PostgreSQL License* http://www.postgresql.org/about/licence/

Szabad és nyílt forrású relációs adatbázis-kezelő rendszerek (3)

- *SQLite* https://www.sqlite.org/
  - Fejlesztő: D. Richard Hipp
  - Platform: Android, Linux, macOS, Windows
  - Programozási nyelv: C
  - Licenc: közkincs
  - Használat módja: beágyazott (*embedded*)
  - Felhasználások: *Well-Known Users of SQLite* https://www.sqlite.org/famous.html
    - Android, Chrome, Dropbox, Firefox, Windows 10, …

Szabad és nyílt forrású relációs adatbázis-kezelő rendszerek (4)

- *H2* http://www.h2database.com/
  - Platform: Java
  - Programozási nyelv: Java
  - Licenc: *Mozilla Public License 2.0/Eclipse Public License 1.0*
  - Használat módja: beágyazott (*embedded*), szerver

Szabad és nyílt forrású relációs adatbázis-kezelő rendszerek (5)

- *HSQLDB* http://hsqldb.org/
  - Fejlesztő: *The HSQL Development Group*
  - Platform: Java
  - Programozási nyelv: Java
  - Licenc: *BSD License* http://hsqldb.org/web/hsqlLicense.html
  - Használat módja: beágyazott (*in-process*), szerver (többféle)

Szabad és nyílt forrású relációs adatbázis-kezelő rendszerek (6)

- *Apache Derby* https://db.apache.org/derby/
  - Fejlesztő: *Apache Software Foundation*
  - Platform: Java
  - Programozási nyelv: Java
  - Licenc: *Apache License 2.0*
  - Használat módja: beágyazott (*embedded*), szerver
  - A *Java DB* az *Apache Derby* egy módosítás nélküli kiadása. http://docs.oracle.com/javadb/
    - Rendelkezésre állás: a JDK része, a JDK installáció `db/` alkönyvtára tartalmazza (JDK 6, JDK 7, JDK 8).
    - A JDK 9 már nem tartalmazza, lásd: *Java Platform, Standard Edition – Oracle JDK 9 Migration Guide – Removed Tools and Components* https://docs.oracle.com/javase/9/migrate/toc.htm#GUID-12237744-E23B-42F8-8892-26BA1BDD63F2

JDBC meghajtók rendelkezésre állása (1)

- *Oracle Database*:
  - *JDBC and Universal Connection Pool (UCP)* https://www.oracle.com/technetwork/database/application-development/jdbc/downloads/index.html
  - *Get Oracle JDBC drivers and UCP from Oracle Maven Repository (without IDEs)* https://blogs.oracle.com/dev2dev/get-oracle-jdbc-drivers-and-ucp-from-oracle-maven-repository-without-ides

JDBC meghajtók rendelkezésre állása (2)

- A központi Maven tárolóban:

Termék	Maven koordináták
Microsoft SQL Server	`com.microsoft.sqlserver:mssql-jdbc:7.1.3.jre11-preview`
MySQL	`mysql:mysql-connector-java:8.0.13`
PostgreSQL	`org.postgresql:postgresql:42.2.5`
SQLite	`org.xerial:sqlite-jdbc:3.25.2`
H2	`com.h2database:h2:1.4.197`
HSQLDB	`org.hsqldb:hsqldb:2.4.1`
Apache Derby (Java DB)	`org.apache.derby:derbyclient: 10.14.2.0` `org.apache.derby:derby:10.14.2.0`

JDBC meghajtó osztályok

- A JDBC megfelelést a `java.sql.Driver` interfész `jdbcCompliant()` metódusa, a JDBC verziót a `java.sql.DatabaseMetaData` interfész `getJDBCMajorVersion()` és `getJDBCMinorVersion()` metódusa szolgáltatja.

Termék	JDBC meghajtó osztály	JDBC megfelelés	JDBC verzió
Oracle Database	`oracle.jdbc.OracleDriver`	igen	4.2
Microsoft SQL Server	`com.microsoft.sqlserver.jdbc.SQLServerDriver`	igen	4.3
MySQL	`com.mysql.jdbc.Driver`	nem	4.2
PostgreSQL	`org.postgresql.Driver`	nem	4.2
SQLite	`org.sqlite.JDBC`	nem	2.1
H2	`org.h2.Driver`	igen	4.0
HSQLDB	`org.hsqldb.jdbc.JDBCDriver`	igen	4.2
Apache Derby (Java DB)	`org.apache.derby.jdbc.ClientDriver` `org.apache.derby.jdbc.EmbeddedDriver`	igen	4.2

JDBC kliensek

- Szabad és nyílt forrású szoftverek:
  - *DBeaver Community Edition* (operációs rendszer: Linux, macOS, Windows; licenc: *Apache License 2.0*) https://dbeaver.jkiss.org/
    - Eclipse-alapú asztali alkalmazás.
  - *JSqsh* (operációs rendszer: Linux, Windows; licenc: *Apache License 2.0*) https://github.com/scgray/jsqsh
  - *SQLLine* (operációs rendszer: Linux, macOS, Windows; licenc: *New BSD License*) https://github.com/julianhyde/sqlline
    - Parancssori eszköz.
  - *SQuirreL SQL Client* (operációs rendszer: Linux, macOS, Windows; licenc: LGPLv2.1) http://squirrel-sql.sourceforge.net/
  - …

IDE támogatás

- **Eclipse**:
  - *Eclipse Data Tools Platform (DTP)* (licenc: *Eclipse Public License v1.0*) https://www.eclipse.org/datatools/
    - Lásd: *Window → Show View → Data Management → Data Source Explorer*
- **IntelliJ IDEA**:
  - Lásd: *View → Tool Windows → Database* https://www.jetbrains.com/help/idea/working-with-the-database-tool-window.html
    - Csak az *Ultimate* kiadásban áll rendelkezésre!
- **NetBeans**:
  - Lásd: *Window → Services → Databases* https://netbeans.org/features/ide/database.html

További olvasnivaló

- *The Java Tutorials: Trail: JDBC Database Access*
  https://docs.oracle.com/javase/tutorial/jdbc/