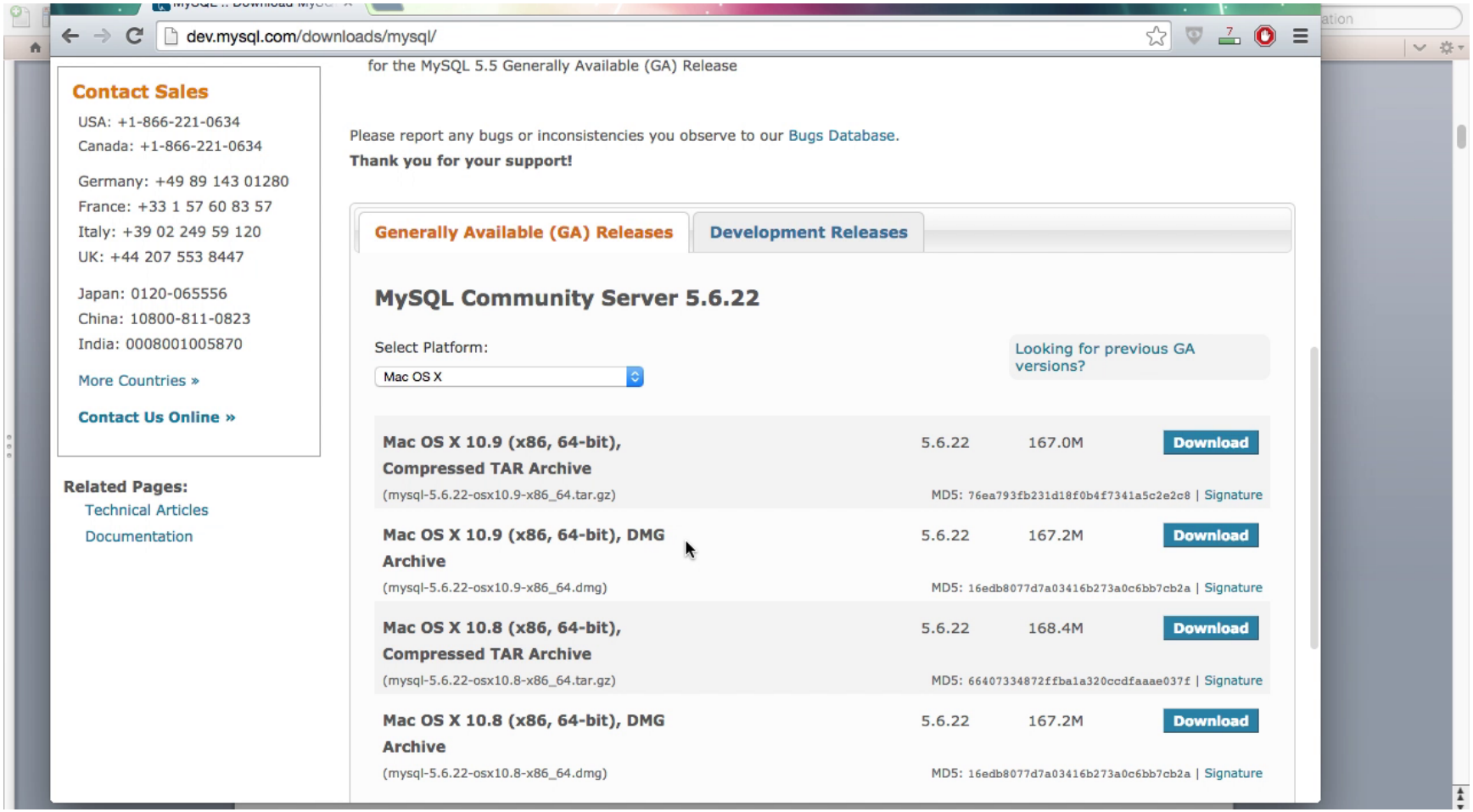
DBMS: MySQL & Hibernate

CORSO DI BASI DI DATI 2015/2016

Indice dei contenuti

- **MySQL**: come configurare il Database
- **MySQL Workbench**: Progettare il database relazionale in modo visuale
- **Hibernate**: Persistency Framework per qualsiasi RDMBS (modulare tramite drivers). Effettua “Object-Relational Mapping”.

1) MySQL


dev.mysql.com/downloads/mysql/
for the MySQL 5.5 Generally Available (GA) Release
Contact Sales
USA: +1-866-221-0634
Canada: +1-866-221-0634
Germany: +49 89 143 01280
France: +33 1 57 60 83 57
Italy: +39 02 249 59 120
UK: +44 207 553 8447
Japan: 0120-065556
China: 10800-811-0823
India: 0008001005870
More Countries »
Contact Us Online »
Related Pages:
Technical Articles
Documentation
Please report any bugs or inconsistencies you observe to our Bugs Database.
Thank you for your support!
Generally Available (GA) Releases
Development Releases
MySQL Community Server 5.6.22
Select Platform:
Mac OS X
Looking for previous GA versions?
Mac OS X 10.9 (x86, 64-bit), Compressed TAR Archive
(mysql-5.6.22-osx10.9-x86_64.tar.gz)
5.6.22
167.0M
Download
MD5: 76ea793fb231d18f0b4f7341a5c2e2c8 | Signature
Mac OS X 10.9 (x86, 64-bit), DMG Archive
(mysql-5.6.22-osx10.9-x86_64.dmg)
5.6.22
167.2M
Download
MD5: 16edb8077d7a03416b273a0c6bb7cb2a | Signature
Mac OS X 10.8 (x86, 64-bit), Compressed TAR Archive
(mysql-5.6.22-osx10.8-x86_64.tar.gz)
5.6.22
168.4M
Download
MD5: 66407334872ffba1a320ccdfaaae037f | Signature
Mac OS X 10.8 (x86, 64-bit), DMG Archive
(mysql-5.6.22-osx10.8-x86_64.dmg)
5.6.22
167.2M
Download
MD5: 16edb8077d7a03416b273a0c6bb7cb2a | Signature

MySQL: configurazione

- In MacOS, dopo l'installazione il PATH potrebbe non essere aggiornato per accedere al client MySQL.

export PATH=${PATH}:/usr/local/mysql/bin

- Quindi accedete al client con il seguente comando, accedendo con la password di default.

mysql -u root -p

- A questo punto dovrete cambiare la password come prima operazione:

SET PASSWORD = PASSWORD('pwd');

Accesso e Creazione DB

- **show databases;**

  Visualizza l'intero set di database memorizzati.

- **create database test;**

  Crea un nuovo database nuovo di nome test

- **use test;**

  Accede al database di nome test

- **show tables;**

  Mostra l'elenco delle tabelle del database corrente

Interrogazione tabella – Sintassi Completa

```
SELECT
    [ALL | DISTINCT | DISTINCTROW ]
      [HIGH_PRIORITY]
      [STRAIGHT_JOIN]
      [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
      [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
    select_expr [, select_expr ...]
    [FROM table_references
    [WHERE where_condition]
    [GROUP BY {col_name | expr | position}
      [ASC | DESC], ... [WITH ROLLUP]]
    [HAVING where_condition]
    [ORDER BY {col_name | expr | position}
      [ASC | DESC], ...]
    [LIMIT {[offset,] row_count | row_count OFFSET offset}]
    [PROCEDURE procedure_name(argument_list)]
    [INTO OUTFILE 'file_name' export_options
      | INTO DUMPFILE 'file_name'
      | INTO var_name [, var_name]]
    [FOR UPDATE | LOCK IN SHARE MODE]]
```

Creazione Indice - Sintassi

CREATE [UNIQUE|FULLTEXT|SPATIAL] INDEX ***index_name***
    [***index_type***]
    ON ***tbl_name*** (***index_col_name***,...)
    [***index_type***]

index_col_name:
    col_name [(***length***)] [ASC | DESC]

index_type:
    USING {BTREE | HASH}

Creazione Indice

```
CREATE INDEX indexprezzo
ON negozio (prezzo)
```

I permessi

Permesso	Istruzioni
ALL	tutte esclusa GRANT
ALTER	ALTER TABLE
CREATE	CREATE TABLE
CREATE TEMPORARY TABLES	CREATE TEMPORARY TABLE
CREATE VIEW	CREATE VIEW
DELETE	DELETE
DROP	DROP TABLE
INDEX	CREATE INDEX, DROP INDEX
INSERT	INSERT
LOCK TABLES	LOCK TABLES
SELECT	SELECT
SHOW VIEW	SHOW CREATE VIEW
UPDATE	UPDATE
USAGE	nessuna
GRANT OPTION	GRANT, REVOKE

Creazione utente

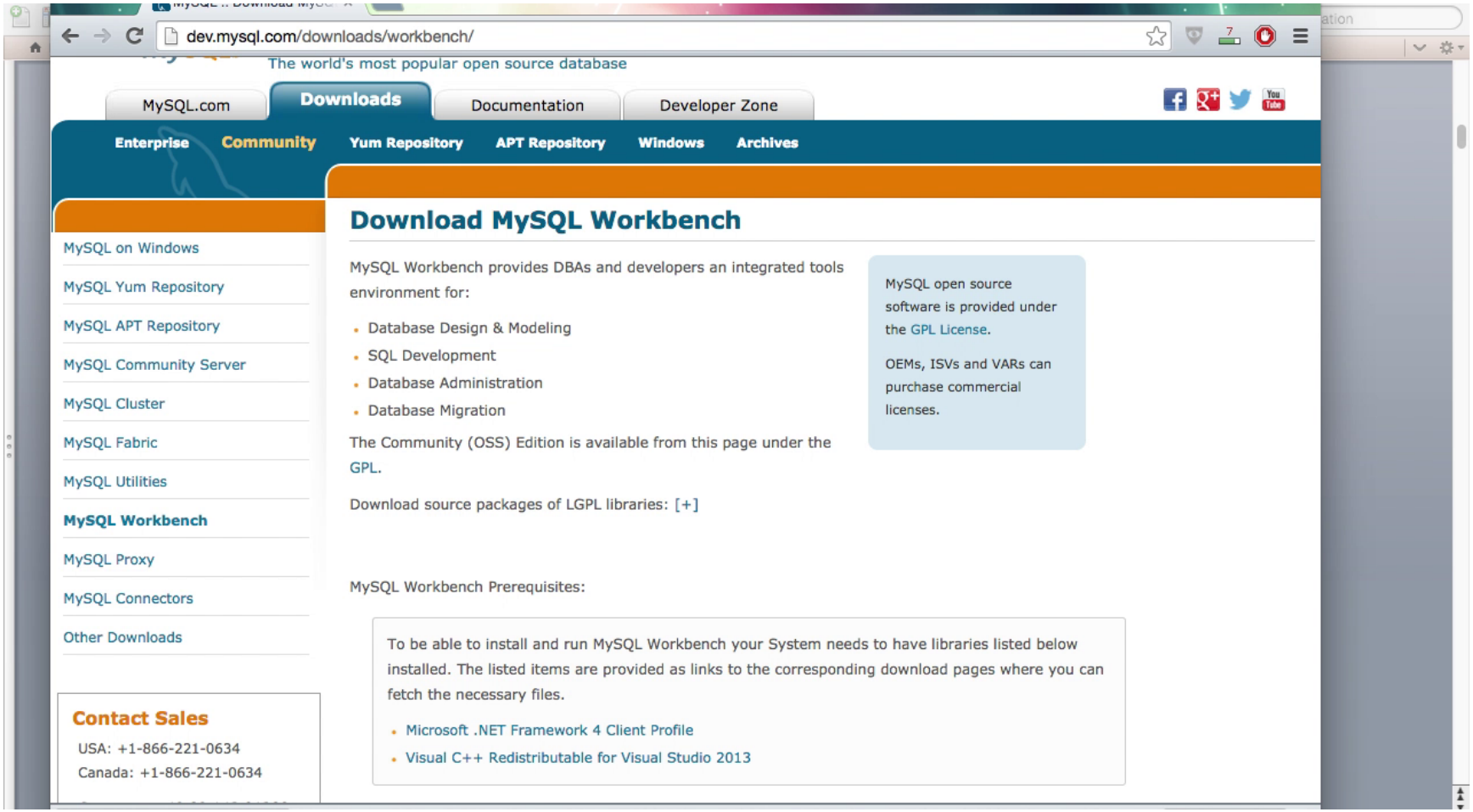
create user ‘pippo’@’localhost’ identified by ‘password_in_chiaro’;

grant all privileges on negozio.* to pippo@localhost ;

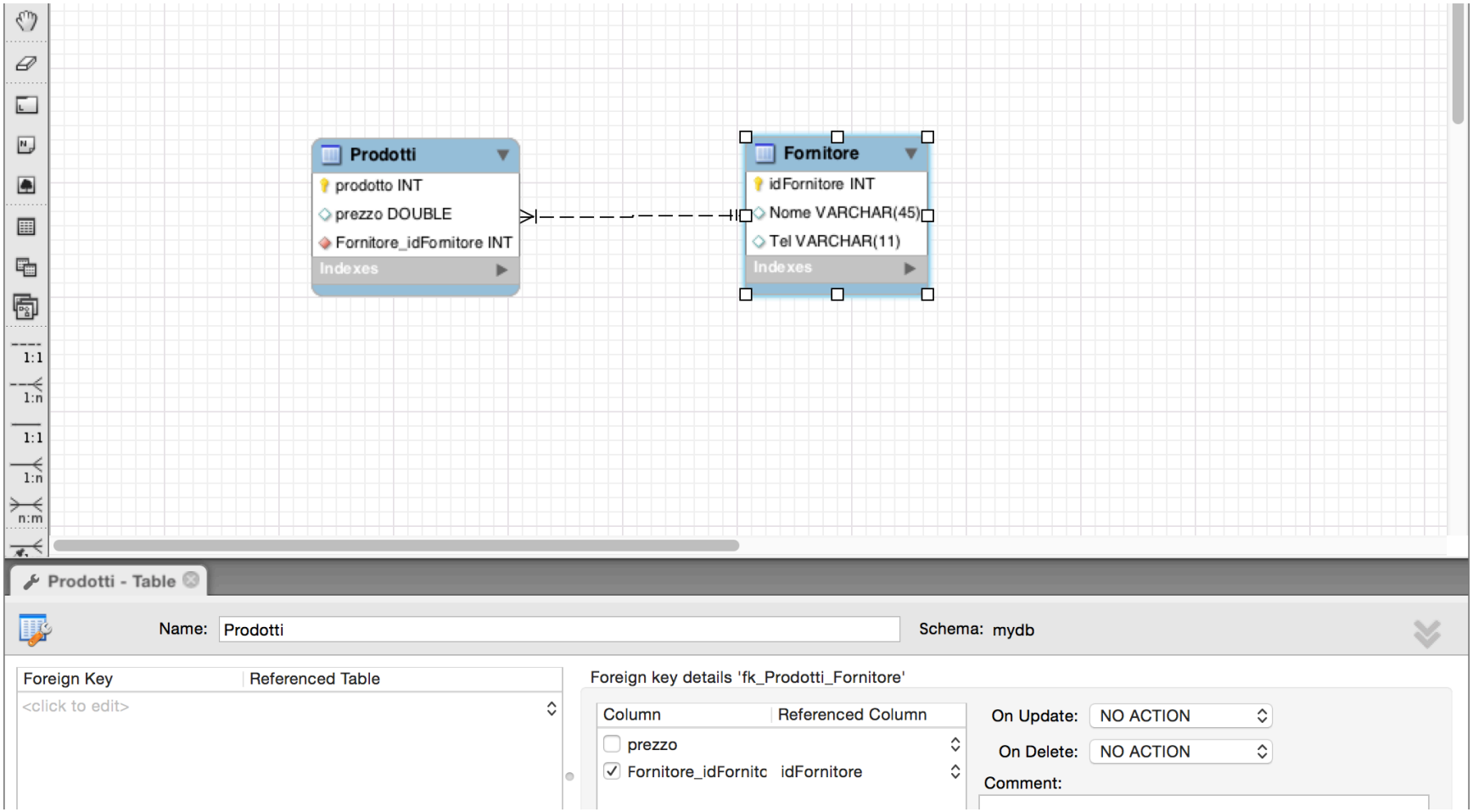
show grant for pippo@localhost;

revoke all privileges on negozio.* FROM pippo@localhost

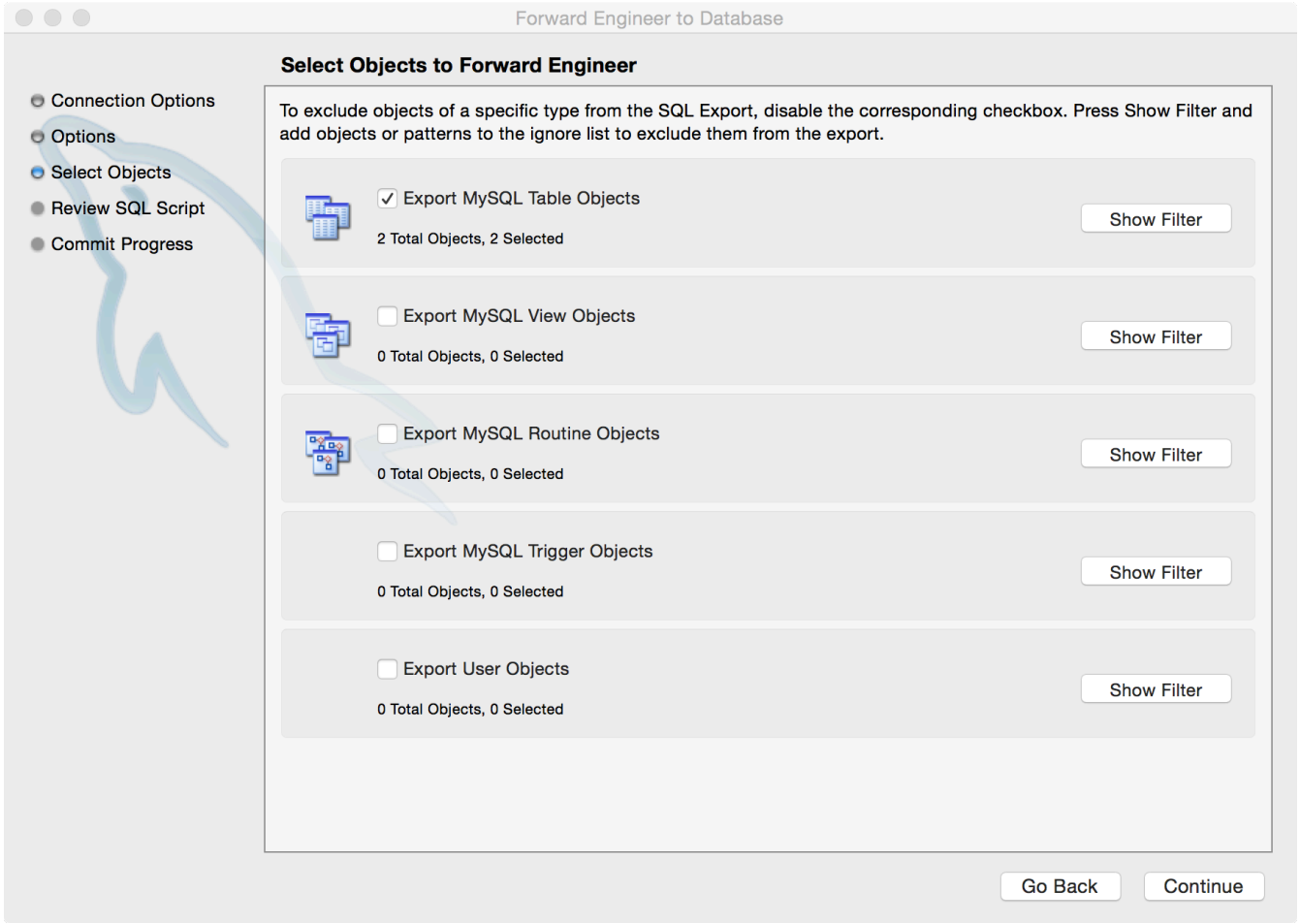
2) MySQL Workbench



FOREIGN Keys in MySQL WB



Forward Engineering in MySQL WB



3) Hibernate

- Un *framework per la gestione della persistenza* consente di mappare direttamente una tabella in un oggetto, istanza di una classe (**O-R mapping service**).
- Non è necessario scandire il database. Il risultato popolerà direttamente una collezione di oggetti.
- Suggerimento: utilizzare "**Hibernate**" per velocizzare il processo.di programmazione (che non è l'obiettivo principale del progetto)
- Java: l'IDE consigliato è **IntelliJ IDEA 15** [**https://www.jetbrains.com/student/**]
- Il progetto d'esempio è su GitHub (https://github.com/jackbergus/javahibernateexample)

Java: Maven (pom.xml)

- **Apache Maven** è un software per la gestione di progetti Java e *build automation*.
- **Apache Maven** usa un costrutto conosciuto come *Project Object Model* (POM); un file XML (pom.ml) che descrive le dipendenze fra il progetto e le varie versioni di librerie necessarie nonché le dipendenze fra di esse.
  - In questo caso si inseriscono le dipendenze per Hibernate, per il driver di accesso al server MySQL, e la libreria esterna per la gestione delle *transactions*.

```
<dependencies>

    <!-- Adding the hibernate Dependencies -->
    <dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate</artifactId>
        <version>3.2.6.ga</version>
    </dependency>
    <dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-annotations</artifactId>
        <version>3.3.1.GA</version>
    </dependency>

    <!-- Adding the mysql driver for hibernate -->
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>5.1.6</version>
    </dependency>

    <!-- java.lang.NoClassDefFoundError: javax/transaction/Synchronization -->
    <dependency>
        <groupId>javax.transaction</groupId>
        <artifactId>jta</artifactId>
        <version>1.1</version>
    </dependency>

</dependencies>
```

Java: Hibernate

(/src/resources/hibernate.cfg.xml)

Il file di configurazione principale va inserito tra le risorse.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-configuration SYSTEM
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
    <session-factory>
        <!-- Using the MySQL database -->
        <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
        <!-- Specifiying the MySQL Driver, imported from pom.xml -->
        <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>

        <!-- Updates and creates the tables -->
        <property name="hibernate.hbm2ddl.auto">update</property>

        <!-- Assume test is the database name, connect to it -->
        <property name="hibernate.connection.url">jdbc:mysql://localhost/test</property>
        <property name="hibernate.connection.username">root</property>
        <property name="hibernate.connection.password">pwd</property>

        <!-- List of XML mapping files (Employee class)
             The xml configuration for the class should be stored in the resources path in the same configuration path
             of the class -->
        <mapping resource="it/db/bergamigiacomo/mysql/Employee.hbm.xml"/>

    </session-factory>
</hibernate-configuration>
```

Da rimuovere se non si vogliono creare le tabelle automaticamente.

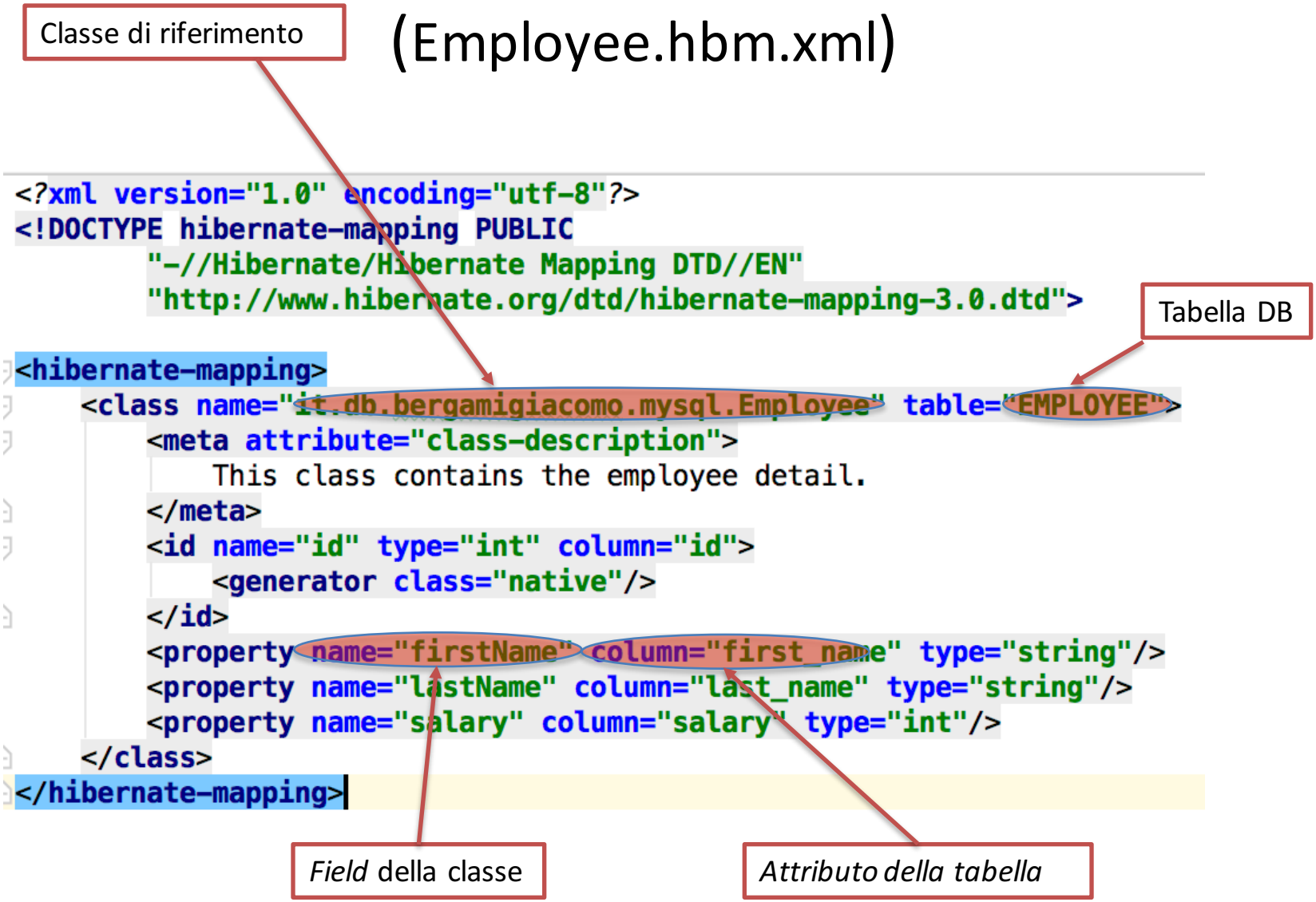
Java: Hibernate
(Employee.java)

- Ad una tabella nel database può corrispondere una classe java **POJO**: i field privati verranno inizializzati da Hibernate e, per ciascuno di questi, dev'essere implementato un getter ed un setter.
- Ad ogni classe java corrisponde un file di configurazione xml (segue)

```java
package it.db.bergamigiacomo.mysql;

public class Employee {
    private int id;
    private String firstName;
    private String lastName;
    private int salary;

    public Employee() {}

    public Employee(String fname, String lname, int salary) {
        this.firstName = fname;
        this.lastName = lname;
        this.salary = salary;
    }
    public int getId() { return id; }
    public void setId( int id ) { this.id = id; }
    public String getFirstName() { return firstName; }
    public void setFirstName( String first_name ) { this.firstName = first_name; }
    public String getLastName() { return lastName; }
    public void setLastName( String last_name ) { this.lastName = last_name; }
    public int getSalary() { return salary; }
    public void setSalary( int salary ) { this.salary = salary; }
}
```

Java: Hibernate

(Employee.hbm.xml)



Java: Hibernate
Popolare una tabella (I)

- Lo scopo principale di queste librerie è quello di evitare SQL injection: conseguentemente la maggior parte delle operazioni utilizza o processori di query o di metodi appositi all'interno di una determinata sessione per eseguire le operazioni sensibili (creazione, eliminazione, aggiornamento)
- Il **singleton** per l'accesso al database è:

```
SessionFactory factory = new Configuration().configure().buildSessionFactory();
```

Java: Hibernate
Popolare una tabella (II)

- Ogni *factory* consente di aprire una *sessione*, all'interno della quale possono essere eseguite delle *transazioni* **tx**, che consentono di eseguire delle operazioni sul database.
- La creazione di una nuova riga corrisponde alla creazione di un nuovo oggetto e del salvataggio dello stesso (**save**)

```
/* Method to CREATE an employee in the database */
public Integer addEmployee(String fname, String lname, int salary){
    Session session = factory.openSession();
    Transaction tx = null;
    Integer employeeID = null;
    try{
        tx = session.beginTransaction();
        Employee employee = new Employee(fname, lname, salary);
        employeeID = (Integer) session.save(employee);
        tx.commit();
    }catch (HibernateException e) {
        if (tx!=null) tx.rollback();
        e.printStackTrace();
    }finally {
        session.close();
    }
    return employeeID;
}
```

Java: Hibernate
Altre operazioni utili

- **session.get(clazz,id);**

  Ottiene l'entry con identificativo **id** dalla classe **clazz** (es. Employee.class).

- **session.update(obj);**

  Aggiorna un oggetto **obj** ottenuto tramite la libreria.

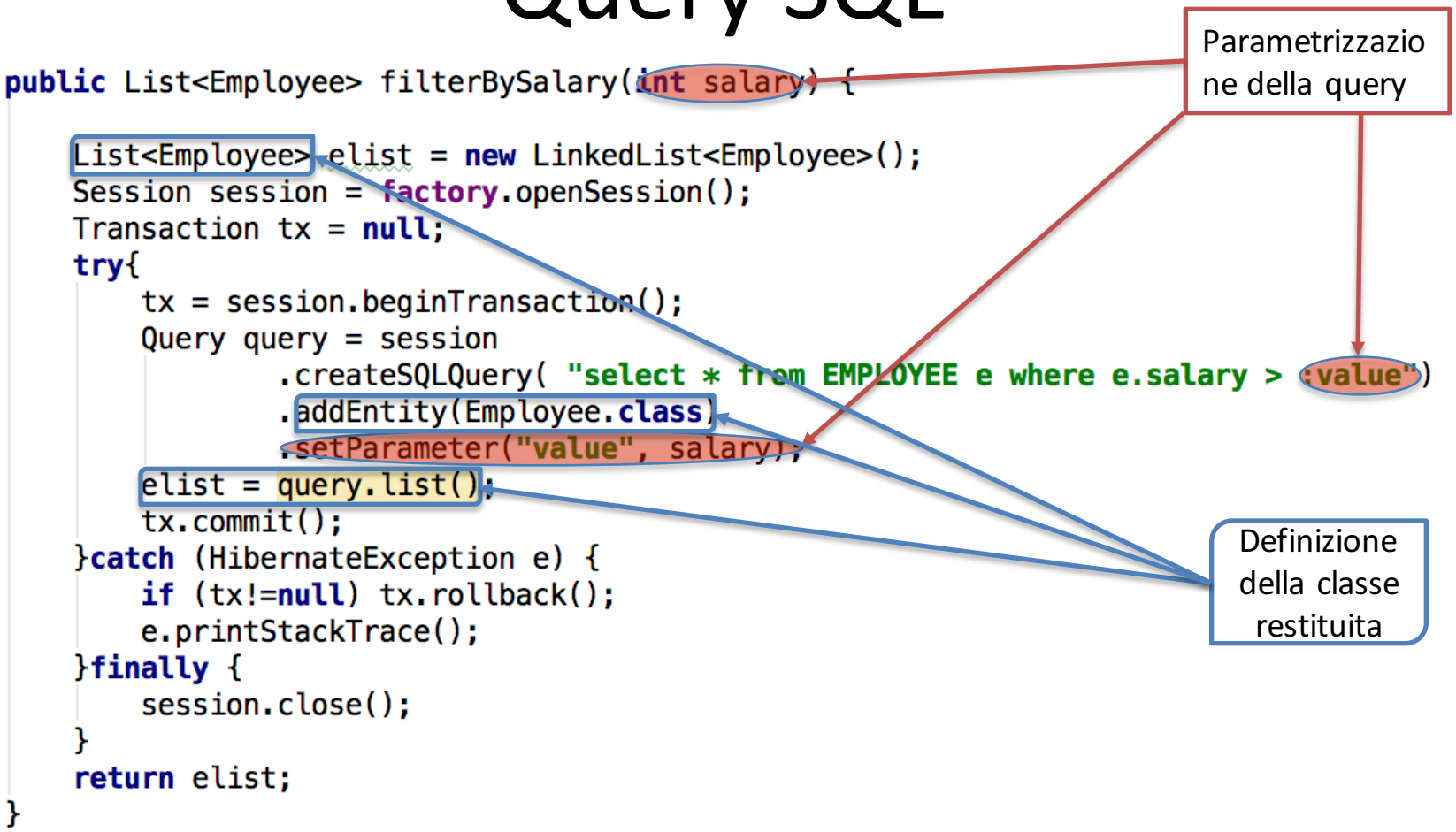
- **session.delete(obj);**

  Rimuove un oggetto **obj** ottenuto tramite la libreria.

- **session.createSQLQuery(string);**

  Crea una query che può essere eseguita. Restituisce una lista di oggetti.

Java: Hibernate Query SQL



Link

- La distribuzione corrente è scaricabile dal sito:

http://dev.mysql.com/downloads/mysql/

- Una documentazione completa è reperibile all'URL:

http://dev.mysql.com/doc/refman/5.7/en/index.html

- Libro on-line sull'utilizzo di Hibernate:

https://www.safaribooksonline.com/library/view/just-hibernate/9781449334369/