Modern Web Development for Java Programmers

Unit 8. Installing and configuring RDBMS. JDBC vs JPA. SQL Mapping framework MyBatis.

Prerequisites: Oracle 11g Express installed.

There’s no Oracle version for MAC - we’ll provide a test server.

Unit 8 Timeline

- Installing and Configuring Oracle 11g Express and SQL Developer, running DDL and DML — 20 min
- Working with Database from IntelliJ IDEA — 10 min
- Walkthrough 1, 2 — 15 min
- Working with JDBC — 15 min
- Walkthrough 3 — 10 min
- JPA — 10 min
- Break — 10 min
- Using MyBatis Framework — 30 min
- Walkthrough 4 — 20 min

Preparing Oracle DB

1. Install Oracle 11g Express
2. Install Oracle SQL Developer
3. In SQL Developer create a connection for SYS admin
4. Create a new user FARATA
5. Connect as the user FARATA
6. Create DB tables Product, Bid, and Auction_Users
7. Populate tables Product, Bid, and Auction_Users with data

Installing Oracle 11g Express

- Download and install Oracle 11g from http://bit.ly/Qmkzpt.
- During the installation you'll need to enter (**and memorize**) the passwords for SYS and SYSTEM accounts.
- Oracle DBMS sevrer starts automatically after install. Start/Stop instructions: http://goo.gl/SwJNtZ

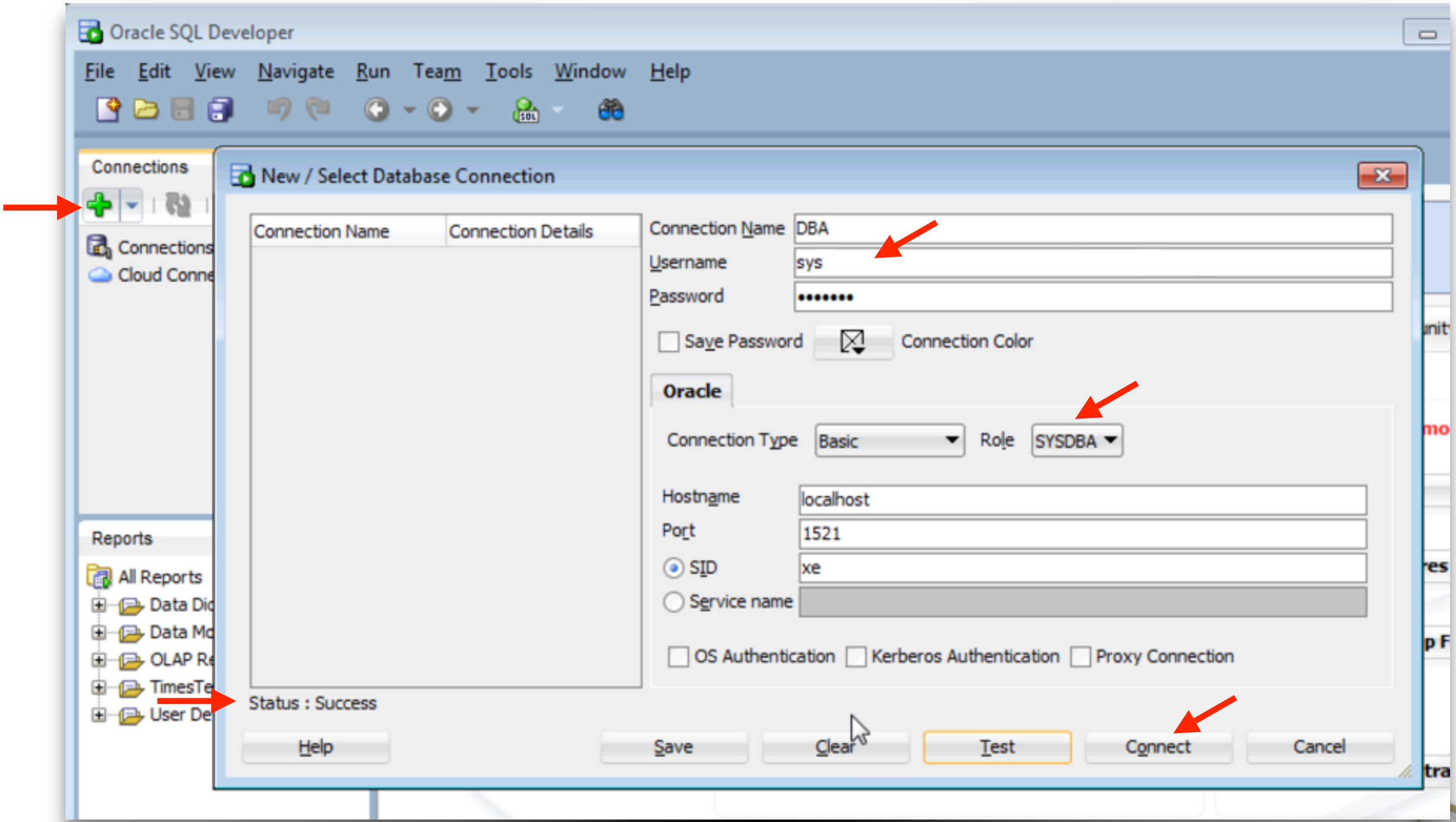
There is no Oracle 11G for Mac OS.
You can download Oracle VM VirtualBox to run it under Win 7.
https://www.virtualbox.org

Installing Oracle SQL Developer

- Download Oracle SQL Developer from http://bit.ly/1bAq4Y7. Unzip it into any folder.
- SQL Developer doesn't support Java 8 yet, so you'll need to install Java 7.
- Run the sqldeveloper.exe located in sqldeveloper subfolder. Specify where your JDK is installed, e.g C:\Program Files\Java\jdk1.7.0_51.

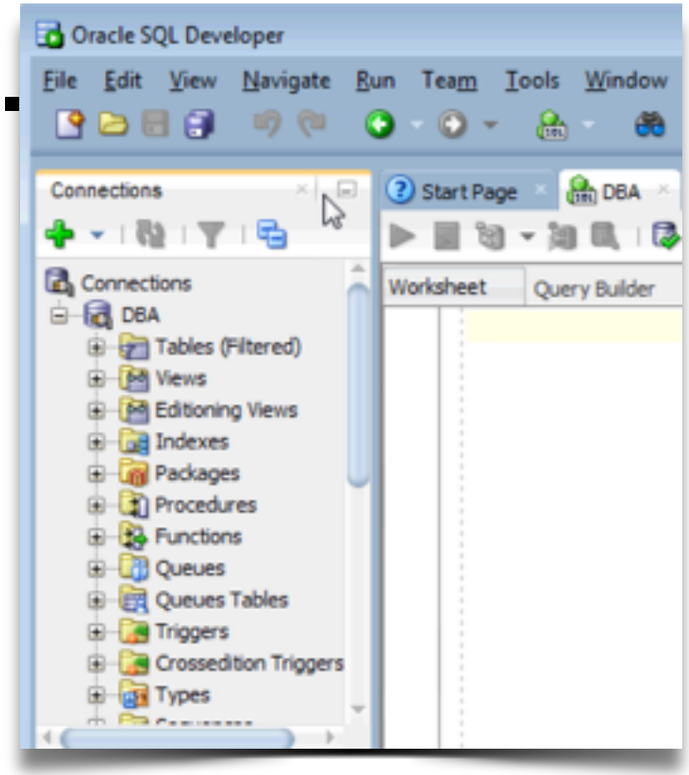
Configuring a new connection for DBA

Hit the green + on the left and configure the DBA connection as shown below.
Click on Test , and then on Connect buttons.

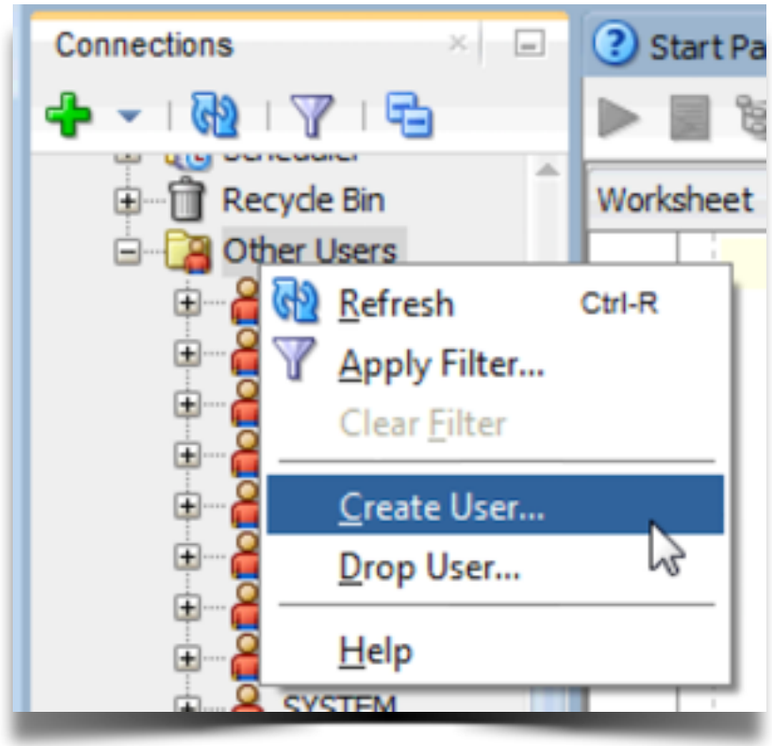


Creating a user and granting roles

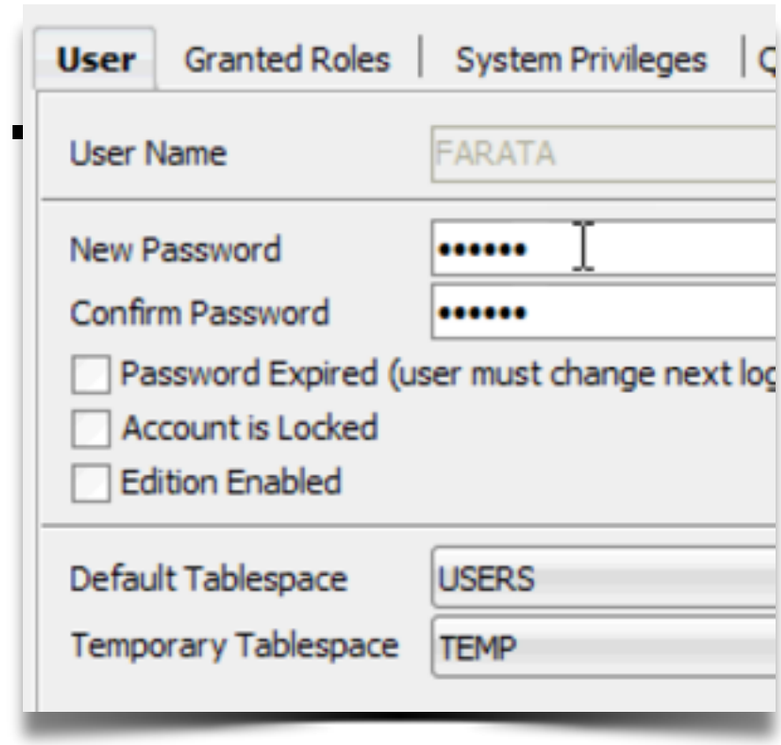
1.



2.



3.

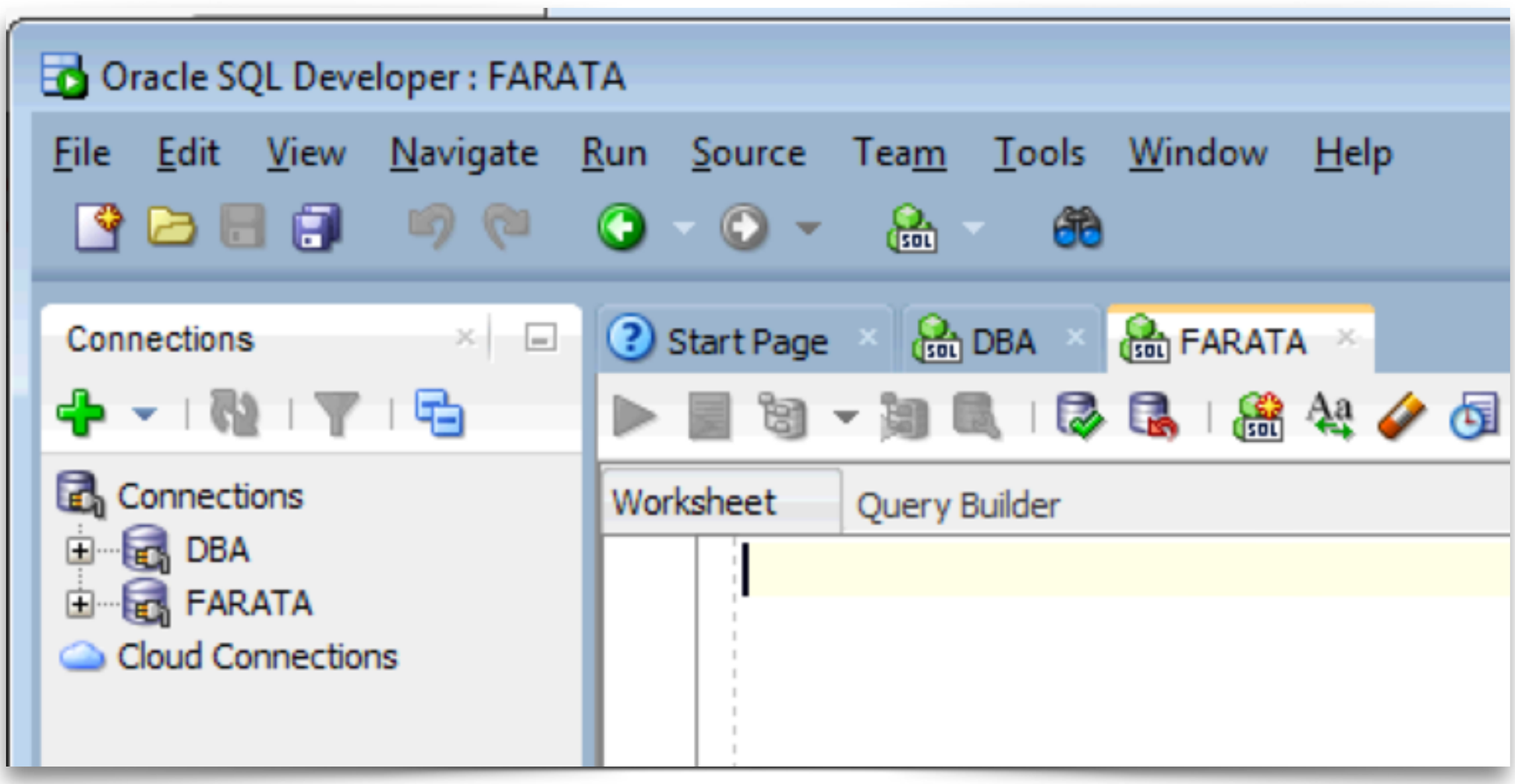


1. Click on DBA,
2. Other Users I Create User.
3. Enter FARATA as uid and pwd, def. tablespace USERS, temp tablespace TEMP.
4. Hit Apply. Close and refresh users.
5. Visit tabs Roles and Priviledges and grant all.
6. In Tab Quotas check the boxes Users in the Unlimited column.
7. Hit Apply in the SQL tab.

Create connection for FARATA user

- Create a new connection for the user FARATA similarly to creating the DBA connection, but leave the role as Default.

- Connect to the database.The window should look like this:



User FARATA has no tables yet

Oracle SQL Developer : FARATA

File Edit View Navigate Run Source Team Tools Window Help

Connections

Connections
- DBA
- FARATA

Cloud Connections

Start Page | DBA | FARATA

Worksheet | Query Builder

```
select table_name
from user_tables;
```

Query Result

SQL | All Rows Fetched: 0 in 0.062 seconds

TABLE_N...

Running DDL

Connections

Connections
- DBA
- FARATA

Cloud Connections

Reports

All Reports
- Data Dictionary Reports
- Data Modeler Reports
- OLAP Reports
- TimesTen Reports
- User Defined Reports

Start Page | DBA | FARATA

Worksheet | Query Builder

```
CREATE TABLE PRODUCT(PRODUCT_ID INTEGER NOT NULL PRIMARY KEY,
                     TITLE VARCHAR(100) NOT NULL,
                     THUMB VARCHAR(100),
                     DESCRIPTION VARCHAR(400),
                     QUANTITY INTEGER NOT NULL,
                     AUCTION_END_TIME DATE NOT NULL,
                     WATCHERS INTEGER,
                     MINIMAL_PRICE NUMBER(8,2) NOT NULL,
                     RESERVED_PRICE NUMBER(8,2) NOT NULL
                     );

CREATE TABLE BID (BID_ID INTEGER NOT NULL PRIMARY KEY,
                  PRODUCT_ID INTEGER NOT NULL,
                  AMOUNT NUMBER NOT NULL,
                  DESIRED_QUANTITY INTEGER,
                  BIDDER_ID INTEGER NOT NULL,
                  BID_TIME DATE NOT NULL,
                  FOREIGN KEY (PRODUCT_ID)
                       REFERENCES PRODUCT(PRODUCT_ID)
                       ON DELETE CASCADE,

                  FOREIGN KEY (BIDDER_ID)
                        REFERENCES AUCTION_USER(USER_ID)
                        ON DELETE CASCADE
                );
COMMIT;
```

Running DML

Start Page | DBA | FARATA

Worksheet | Query Builder

```
INSERT INTO AUCTION_USER VALUES (1, 'alex123', 'alex@gmail.com',1);
INSERT INTO AUCTION_USER VALUES (2, 'sammy777', 'sammy@gmail.com',1);
INSERT INTO AUCTION_USER VALUES (3, 'mary454', 'mary@gmail.com',1);


INSERT INTO PRODUCT VALUES (100, 'Gucci Handbag','Gbag1.png',
                    'The latest model from Gucci made in Chinatown!', 5,
                    TO_DATE('2014/05/15 21:00:00', 'yyyy/mm/dd hh24:mi:ss'),
                    0,450.00,800.00);

INSERT INTO PRODUCT VALUES (101, 'Samsung Earbuds','earbd1.png',
                    'Earbuds are brand new without factory packaging', 100,
                    TO_DATE('2014/05/20 13:00:00', 'yyyy/mm/dd hh24:mi:ss'),
                    0,1.00,2.00);

INSERT INTO BID VALUES(1000,100,300,2,1,SYSDATE);
INSERT INTO BID VALUES(1001,100,450,1,3,SYSDATE);

COMMIT;
```

Configuring IntelliJ IDEA

Walkthrough 1. Connecting to Oracle form IDEA

- Create a new Java project selecting SQL Support checkbox and Oracle as a dbms
- Open menu View | Tool Windows | Database.
- Click on the Plus sign on top to configure new data source selecting Oracle as DBMS.
- If there is a message Driver Files Missing, click on the provided link to download Oracle driver files. You can see Drivers section on the left.
- Enter IP of the host, port, and xe as the database name as shown in the next slide.
- Enter the user's id and password and test the connection (see the next slide for expected output). Press Apply.
- In the Schema & Tables tab load FARATA schema.

Connection from IDEA successful

Data Sources and Drivers

Data Sources
- Oracle - xe@64.135.72.221

Drivers

Name: Oracle - xe@64.135.72.221

Scope: IDE

Database | Schemas & Tables | Advanced

Host: 64.135.72.221

Port: 1521

Database: xe

The following URL will be used to connect, edit to override:

jdbc:oracle:thin:@//64.135.72.221:1521/xe

User: FARATA

Password: ••••••

Configure SSH | Configure SSL | Test Connection

Oracle - xe@64.135.72.221

Database: Oracle/Oracle Database 11g Express Edition Release 11.2.0.2.0 - Production
Driver name: Oracle JDBC driver
Driver version: 11.2.0.3.0
JDBC version: 11.2
Case sensitivity: UPPER (quoted: EXACT)

Connection successful (13 s)

OK

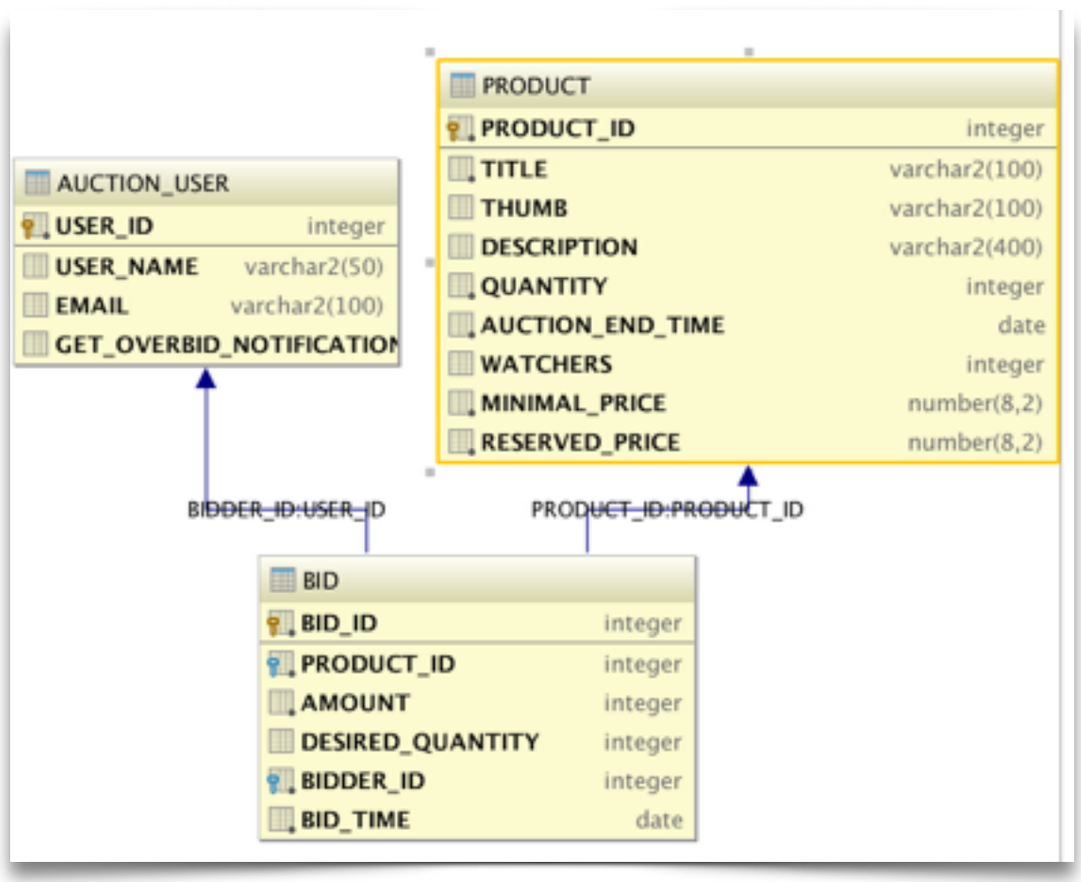
Read-only | Auto sync

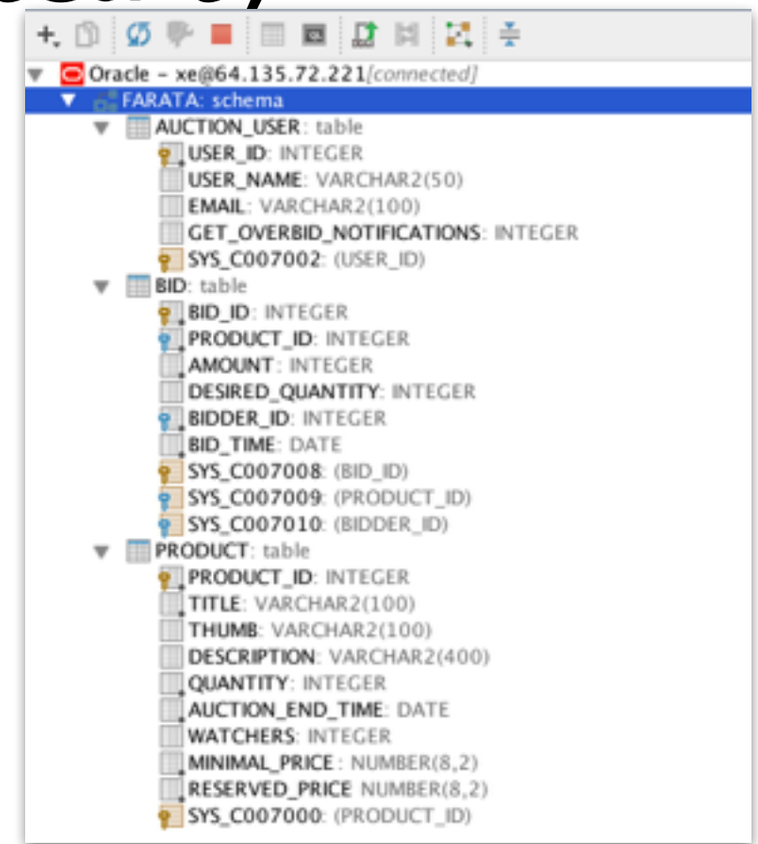
Cancel | Apply | OK

Walkthrough 2 (start)

1. Open Configured Oracle datasource and Farata Schema.

2. Click on the Diagram button.



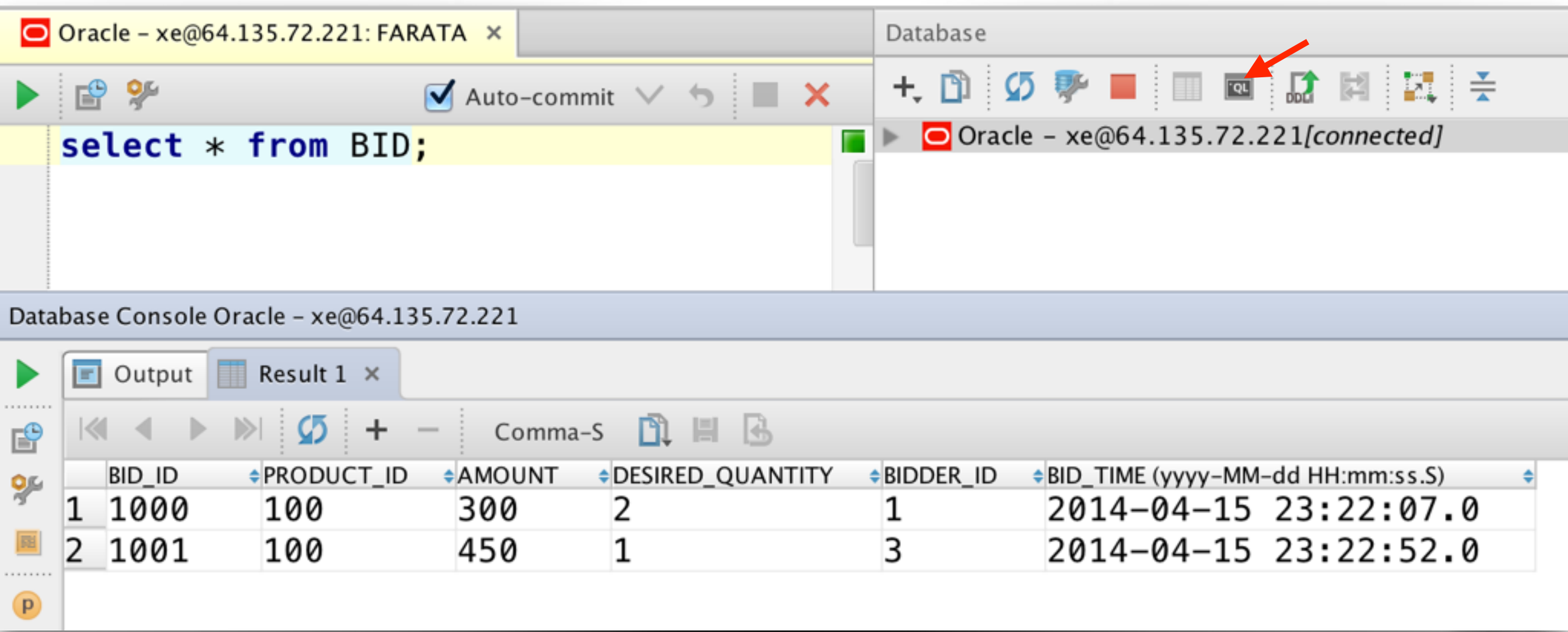


3. Right-click on the table name to open Table Editor

Walkthrough 2 (cont.)

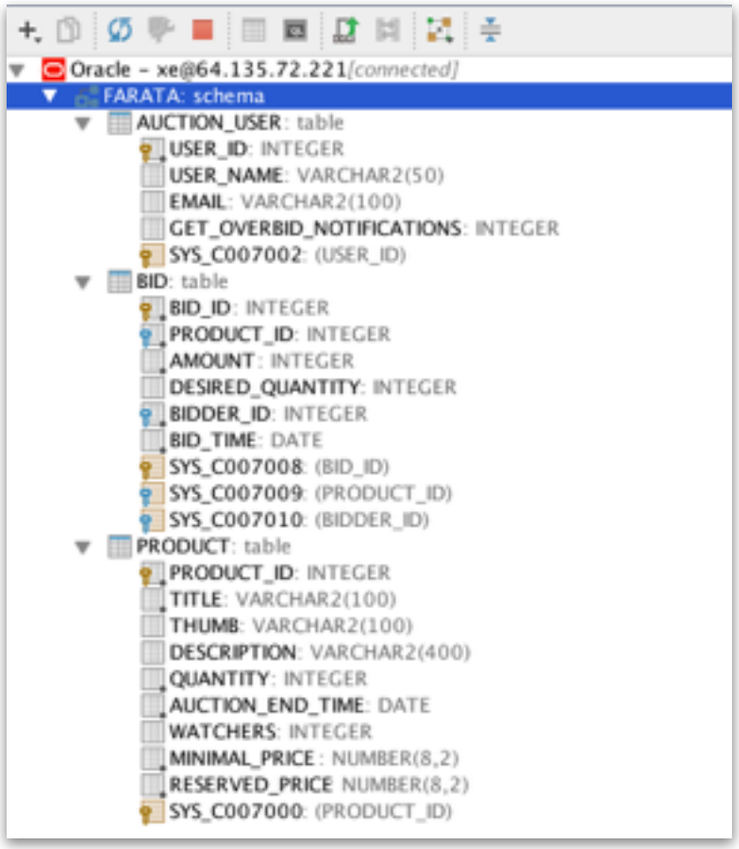
Open the SQL console.

Enter the SQL query and press the Play button. Note that the autocompletion works.

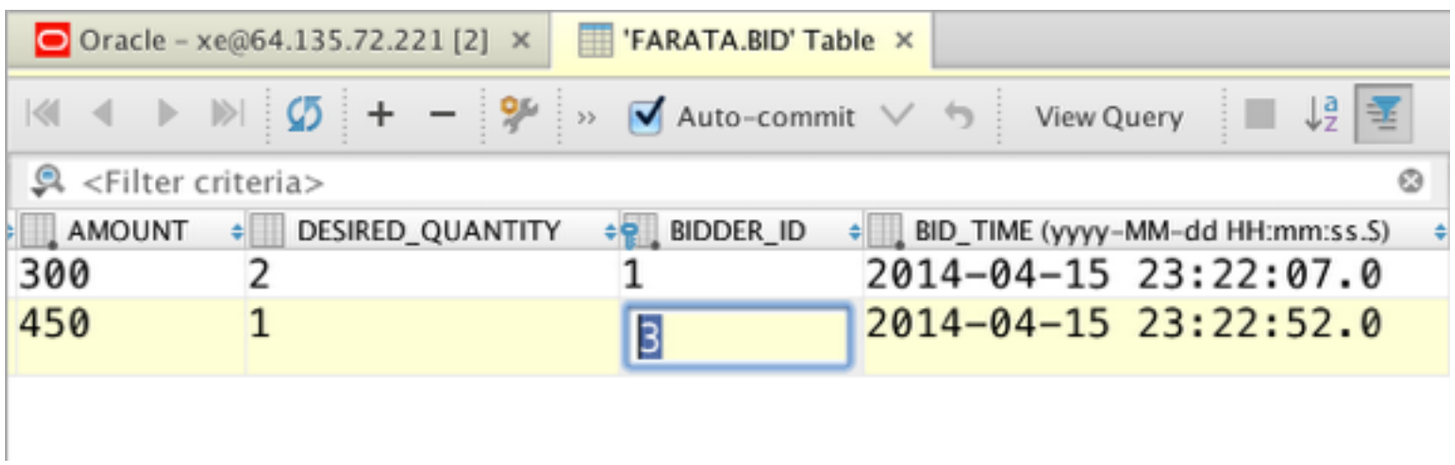


Walkthrough 2 (end)

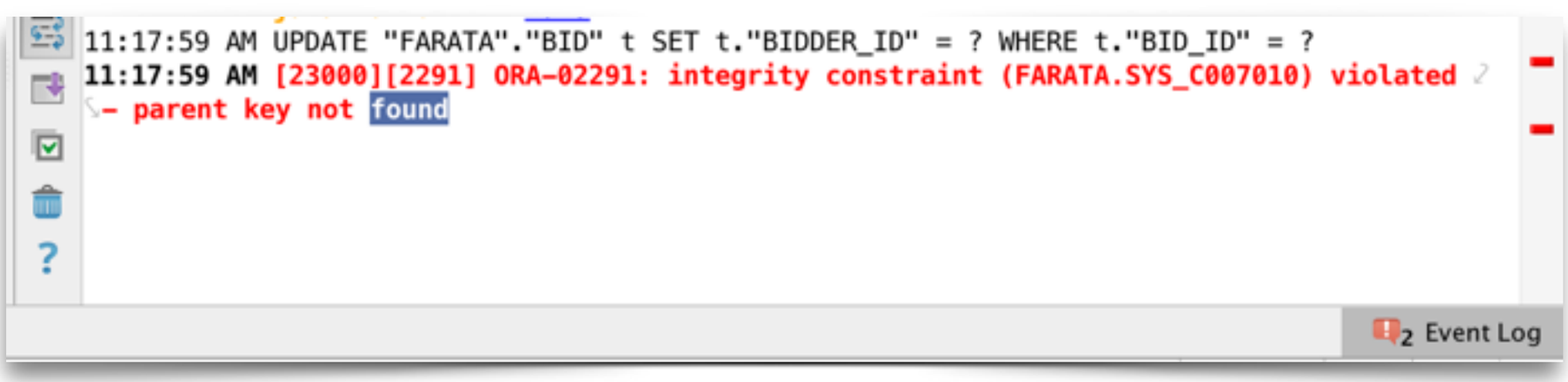
Right-click on the Bid table name to open Table Editor.



Try change the BIDDER_ID to non-existing user id (e.g. 777) and tab out of the field.



The event log shows the error.



Java and JDBC

JDBC driver Types

Type 1 driver is a JDBC-ODBC bridge that enables Java programs to work with the database using ODBC drivers from Microsoft. Windows only.

Type 2 driver: native drivers are wrapped in Java (e.g. Oracle OCI driver). These must be installed on the computer that runs client Java program accessing DBMS.

Type 3 driver consists of two parts: the client portion relays a DBMS independent SQL to middleware server, which then translates it to a specific DBMS protocol by the server portion of the driver.

Type 4 driver is a pure Java thin driver, which comes as a .jar file and performs direct calls to the database server. It does not need any configuration on the client's machine.

Connecting With JDBC `DriverManager`

```
String url = "jdbc:oracle:thin:@//64.135.72.221:1521/xe";
Properties props = new Properties();
props.setProperty("user", "FARATA");
props.setProperty("password", "itsasecret");

try {

    // On the server-side getting connectinon should be done by injecting
    // the DataSource object with @Resource annotation
    Connection conn = DriverManager.getConnection(url,props);

    String sql ="select sysdate as currentTime from dual";

    PreparedStatement preStatement = conn.prepareStatement(sql);

    ResultSet result = preStatement.executeQuery();

    while(result.next()){
            System.out.println("Oracle DBMS returned " +
                    result.getString("currentTime"));
        }
} catch (SQLException e) {
        e.printStackTrace();
}
```

`DataSource` - a Preferred Way to Connect

- Connecting to DBMS is a slow process, it would be very inefficient to perform connect/disconnect for every SQL request.
- Connection pools allow reusing Connection objects.
- The package javax.sql includes the interface `DataSource`, which is an alternative to `DriverManager`.
- JDBC drivers implement this interface, and a `DataSource` is typically pre-configured for a certain number of connections in the pool.
- The `DataSource` interface is typically used on the server side bound to JNDI.

Configuring DataSource in Wildfly

```
<datasources>
    <datasource jndi-name="java:jboss/datasources/ExampleDS" pool-name="ExampleDS">
        <connection-url>jdbc:h2:mem:test;DB_CLOSE_DELAY=-1</connection-url>
        <driver>h2</driver>
        <pool>
            <min-pool-size>10</min-pool-size>
            <max-pool-size>20</max-pool-size>
            <prefill>true</prefill>
        </pool>
        <security>
            <user-name>sa</user-name>
            <password>sa</password>
        </security>
    </datasource>
    <xa-datasource jndi-name="java:jboss/datasources/ExampleXADS" pool-name="ExampleXADS">
        <driver>h2</driver>
        <xa-datasource-property name="URL">jdbc:h2:mem:test</xa-datasource-property>
        <xa-pool>
            <min-pool-size>10</min-pool-size>
            <max-pool-size>20</max-pool-size>
            <prefill>true</prefill>
        </xa-pool>
        <security>
            <user-name>sa</user-name>
            <password>sa</password>
        </security>
    </xa-datasource>
    <drivers>
        <driver name="h2" module="com.h2database.h2">
            <xa-datasource-class>org.h2.jdbcx.JdbcDataSource</xa-datasource-class>
        </driver>
    </drivers>
</datasources>
```

Walkthrough 3

1. Create a new Java class JavaOraDriverManager and copy the source code from the provided class.
2. Copy the URL value from the configured data source in Walkthrough 1 and assign it to the url variable in the code.
3. Download ojdbc7.jar thin JDBC driver for JDK 7 from http://bit.ly/1d4rn1Z , and add it to your IDEA module using Project Stucture | Libraries| +.
4. Run the program - it shoud print the current date.
   If you see the error “No Suitable driver found” go back to step 3.
5. Replace the SQL statement with this one:
   select AUCTION_END_TIME from PRODUCT
7. Modify the loop statement to be result.getString(“AUCTION_END_TIME"));
8. Re-run the program - you should see all auction end times from the database
9. Modify the code to print the product title, desccrription and minimal price

IDEA can autocomplete your SQL. Click inside the SQL string, click on the lightbulb, select Language Injection Settings and Oracle (SQL Files).

Java Persistence API

brief overview

What's JPA

Java Persistence API offers a way for object-relational mapping of Java objects to objects from relational databases.

JPA is created for people who can't master SQL.

JPA Spec is 500 pages long. Time's better spent learning SQL.

To avoid writing boilerplate JDBC code use MyBatis framework that maps Java objects to SQL statements.

You can run from SQL, but you can't hide.

JPA ways of querying DB

- Java Persistence Query Language (JPQL)

  It's used by people who have heard of SQL.

  JPQL is not type-safe and requires casting when retrieving results.

  Dynamic JPQL queries must be parsed every time they are called

- The Java Persistence Criteria API

  It's used by Java nazis who suffer from SQLphobia.

  Criteria API is even more verbose than JPQL, but allows Java racists not learn anything but Java.

Don't trust me? Read Oracle doc: http://goo.gl/VqJoC9

Entity Classes

A Java bean that's marked with @Entity is called *an entity*.

Each entity instance corresponds to a row in a database table.

If you start with an empty database, JPA tools can create database entities based on Java entities.

You can also map Java entities to the existing database tables.

```
@Entity
public class Employee implements Serializable{

  @Id
  @GeneratedValue(strategy=GenerationType.IDENTITY)
  private Long id;

  @NotNull
  @Size(max=10)
  public String firstName;

  @NotNull
  @Size(min=2, max=20)
  public String lastName;

  @Column(name="boss_name")
  public String managerName;

  @OneToMany (mappedBy = "employee")
  public List<Address> adresses = new ArrayList<Address>();

}
```

JPQL

- JPQL is similar to SQL
- SQL operates on RDBMS objects, but JPQL operates on Java entities

```
SELECT e.managerName,
FROM Employee AS e
WHERE e.lastName='Smith'

SELECT e.firstName, e.lastName
FROM Employee AS e
WHERE e.companyPhone.model='iPhone'

SELECT e FROM Employee AS e

SELECT DISTINCT e
FROM Employee AS e JOIN e.addresses as a
WHERE a.city = 'New York'
```

Entity Manager

- EntityManager executes all JPA requests to communicate with RDBMS
- Each instance ot EntityManager is associated with a set of entities (a.k.a. persistence unit).

```
@PersistenceContext EntityManager em;  //
injection of Entity Manager

Employee employee = em.find(Employee.class,
1234); // find an employee with id=1234

@Resource UserTransaction userTransaction;
…
Employee newEmployee = new Employee();
newEmployee.firstName=”Mary”;
newEmployee.lastName=”Thompson”;
…
  userTransaction.begin();
  em.persist(newEmployee);
  em.remove(oldEmployee);

 userTransaction.commit();
```

Querying with JPQL

- `EntityManager` executes all JPA requests to communicate with RDBMS
- Each instance ot `EntityManager` is associated with a set of entities (a.k.a. persistence unit).

```
EntityManager em;
List employees;
…
employees = em.createQuery(
"SELECT e.managerName FROM Employee AS e
WHERE e.firstName='Mary' AND
e.lastName='Thompson'").getResultList();
```

```
EntityManager em;
List employees;

String fName = "Mary";
String lName = "Thompson";
…
employees = em.createQuery("SELECT e.managerName FROM Employee AS e
WHERE    e.firstName= :fname AND lastName= :lname")
   .setParameter("lname", lastName)
   .setParameter("fname", firstName)
   .getResultList();
```

Querying with Criteria API

The following Criteria query is taken from Oracle doc: http://goo.gl/7kMW45.

It returns all instances of the Pet entity in data source:

```
EntityManager em = …;

CriteriaBuilder cb = em.getCriteriaBuilder();
CriteriaQuery<Pet> cq = cb.createQuery(Pet.class);

Root<Pet> pet = cq.from(Pet.class);

cq.select(pet);

TypedQuery<Pet> q = em.createQuery(cq);

List<Pet> allPets = q.getResultList();
```

The equivalent JPQL query is

```
SELECT p
FROM Pet p
```

Working with MyBatis Framework

MyBatis

- SQL mapper framework
  - POJOs and Collections (Maps, Lists) define a result
- XML or annotations for configuration
- Integration with widely used technologies (Spring, CDI, Guice, etc)

Why it’s important

- MyBatis is not ORM
- SQL is a first class citizen
- DSL for dynamic SQL

Configuration - SQL Maps

- XML
- Annotations
- Annotation + XML

Java vs XML

- Java interfaces define the contract
- Annotations with SQL statements
- Implementation (in runtime) provide by XML or annotations

Additional reading

- MyBatis http://mybatis.github.io/mybatis-3/
- MyBatis generator http://mybatis.github.io/generator/index.html
- MyBatis CDI module http://mybatis.github.io/cdi/
- MyBatis Spring http://mybatis.github.io/spring/

Walkthrough 4

- `ProductMapper.java`
- `ProductMapper.xml`
- `Product.java`
- `Application.java`

Homework

- Configure Oracle DataSource object in Wildfly JNDI as described in the online doc “DataSource Configuration” http://goo.gl/WXk3i7.
- Incorporate JNDI connection lookup in the Product Rest endpoint.
- Integrate MyBatis and CDI in WildFly
- Add SQL (PreparedStatement or mybatis) to find products in Oracle DBMS in the @Get method implementation of the Product Endpoint.

Additional Materials

- IDEA Database Tools tutorial: http://goo.gl/XX6nVX
- Accessing and manipulating Oracle Data: http://bit.ly/1eF01nC
- Java/Oracle Data Type Mapping: http://bit.ly/QahHMD
- Oracle JDBC FAQ: http://bit.ly/1qEYQ9q
- JPA Tutorial: http://goo.gl/SK49A6
- MyBatis Getting Started: http://goo.gl/Z2VzgC