

1

DB Visual ARCHITECT SQL

Chapter 1 – DB Visual ARCHITECT SQL

DB Visual ARCHITECT SQL (DB-VA SQL) is a full featured SQL development environment for generating the SQL statements and code template for database manipulation. It allows you to edit and execute the generated statements and/or code to manipulate the connected database. DB-VA SQL supports not only the generation of SQL statements and Java code, but also the Java and .NET persistence code to manipulate the database through object-relational mapping. This chapter shows you how to use DB-VA SQL to manipulate the database and assist the project implementation.

In this chapter:

- ◆ Introduction
- ◆ Launching DB Visual ARCHITECT SQL
- ◆ Using the Editor
- ◆ Configuring Database Connection
- ◆ Generating SQL Statement and Java Code for Database Manipulation
- ◆ Generating Persistence Code for Database Manipulation
- ◆ Executing SQL Statements and Code
- ◆ Exporting SQL Statements and Code

Introduction

DB Visual ARCHITECT SQL (DB-VA SQL) is a full featured SQL development environment for most of the popular databases in the market, including Oracle, DB2, Microsoft SQL Server, Sybase Adaptive Server Enterprise, Sybase SQL Anywhere, MySQL, HSQLDB, Cloudscape/Derby and PostgreSQL.

DB-VA SQL enhances the productivity by providing a set of useful features for database development. It not only provides a text editor for editing the SQL statements, Java code, Java persistence code and .NET persistence code which are generated by the other VP's products, but also supports the generation of SQL statements, Java code, Java and .NET persistence code to directly manipulate the database.

Within DB-VA SQL, you can use the generated statements and/or code to access and manipulate the database directly. By executing the statements and/or code, the real-time result is shown. If you are a beginner to use ORM-Persistable class to develop application, you can take the advantage of DB-VA SQL generating the code template to manipulate persistent data with the database by inserting, retrieving, updating and deleting records.

DB-VA SQL can be used within the working environment of the other Visual Paradigm's products; including Visual Paradigm for UML (VP-UML), DB Visual ARCHITECT (DB-VA) and Smart Development Environment (SDE), or in the standalone version.

Using standalone version of DB-VA SQL, you can manipulate the database directly using the ease-to-use graphical user interface to reduce the handling of SQL statements manually.

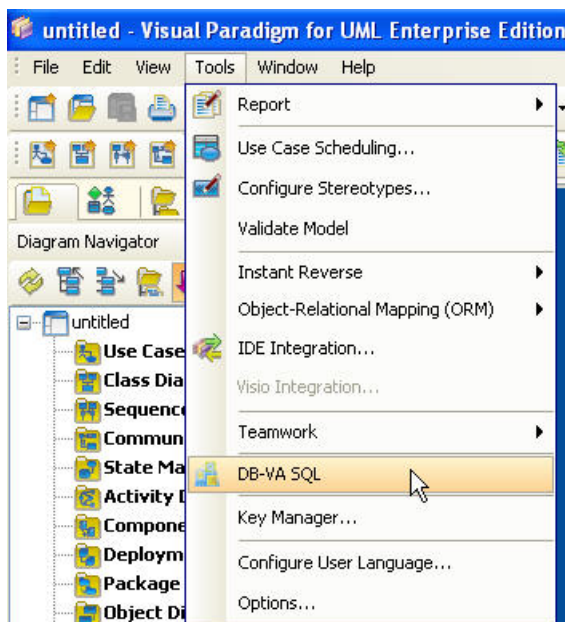
Launching DB Visual ARCHITECT SQL

As DB Visual ARCHITECT SQL (DB-VA SQL) is executable with the working environment of the other Visual Paradigm's products and in standalone version, you can launch the DB-VA SQL in either version.

Within VP-UML environment

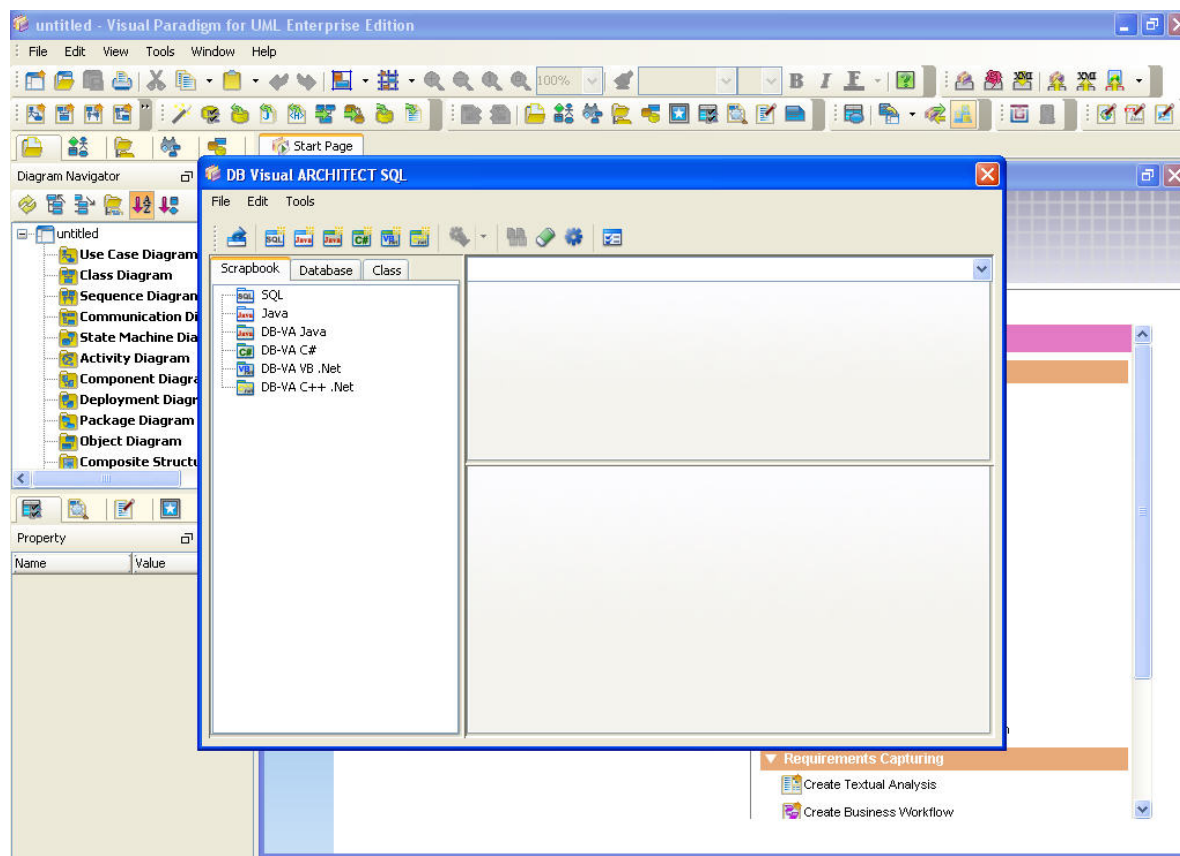
You can launch the DB-VA SQL within VP-UML working environment in one of the two ways:


- On the menu, click **Tools > DB-VA SQL**.



- On the toolbar, click the **DB-VA SQL** icon.

The DB-VA SQL is running on top of the VP-UML working environment.

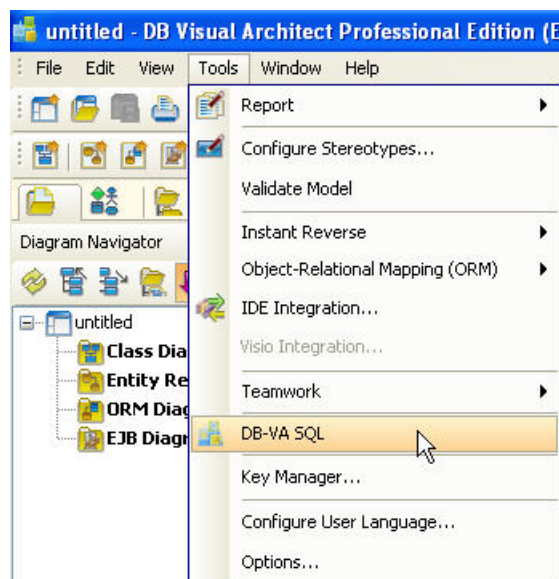


 VP-UML is in the background mode when running the DB-VA SQL inside the VP-UML environment, you must exit the DB-VA SQL to activate the VP-UML environment.

Within DB-VA environment

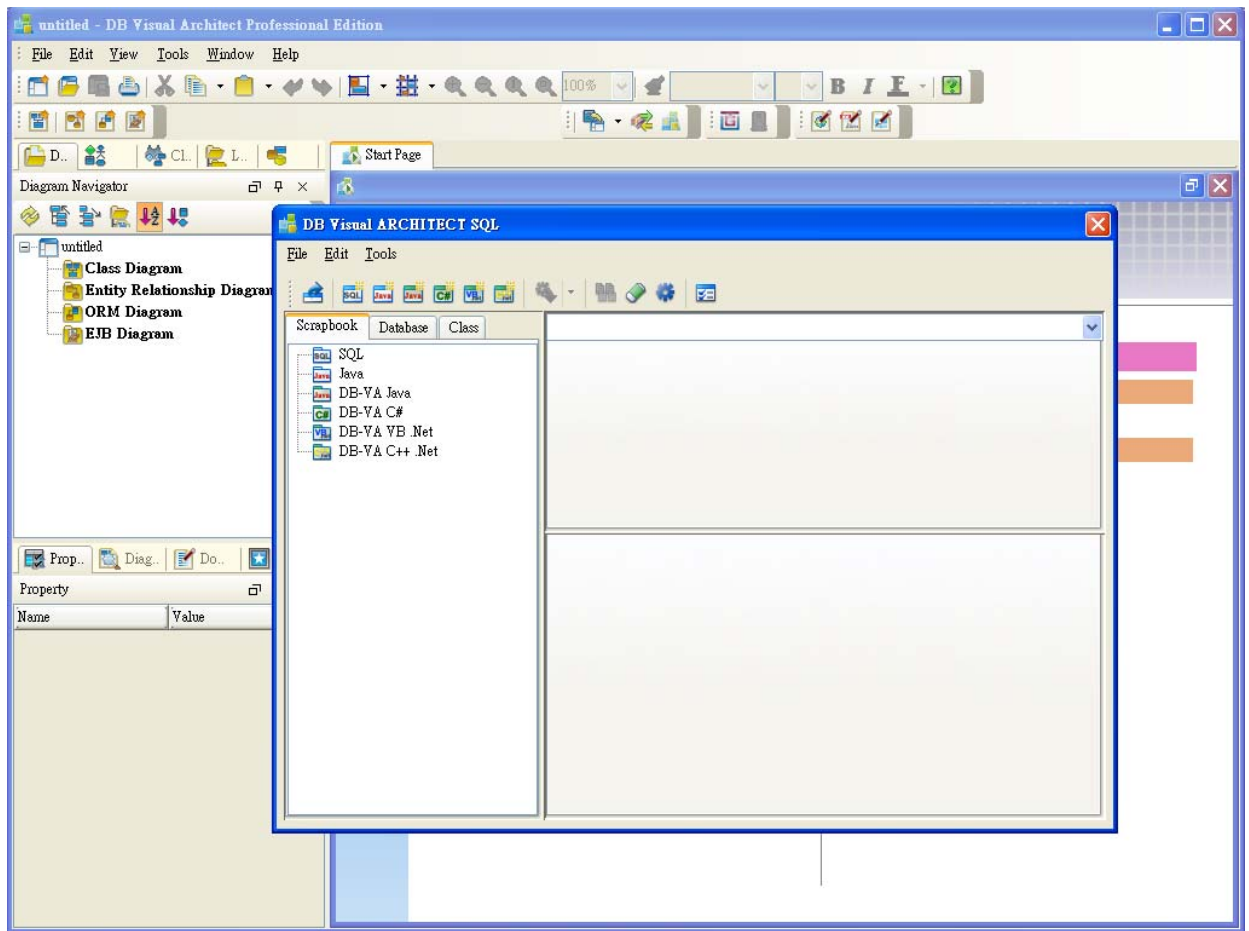
You can launch the DB-VA SQL within DB-VA working environment in one of the two ways:

- On the menu, click **Tools > DB-VA SQL**.



- On the toolbar, click the **DB-VA SQL** icon.

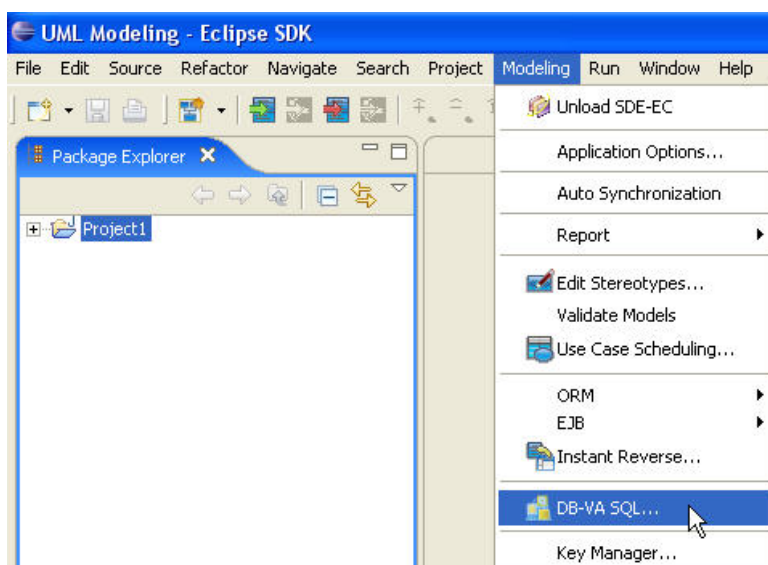
The DB-VA SQL is running on top of the DB-VA working environment.



DB-VA is in the background mode when running the DB-VA SQL inside the DB-VA environment, you must exit the DB-VA SQL to activate the DB-VA environment.

Within SDE environment

You can launch the DB-VA SQL within SDE for Eclipse by clicking **Modeling > DB-VA SQL...** from the menu.



For other SDE:

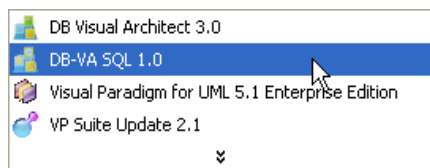
SDE	Method
SDE for JBuilder	From the menu, click Tools > Modeling > DB-VA SQL....
SDE for NetBeans	From the menu, click Modeling > DB-VA SQL....
SDE for IntelliJ IDEA	From the menu, click Modeling > DB-VA SQL....
SDE for JDeveloper	From the menu, click Model > DB-VA SQL....
SDE for WebLogic Workshop	From the menu, click Modeling > DBVA SQL....
SDE for Visual Studio	From the menu, click Modeling > DBVA SQL....

 SDE is in the background mode when running the DB-VA SQL inside the SDE environment, you must exit the DB-VA SQL to activate the SDE environment.

Standalone Version

You can launch the standalone version of DB-VA SQL in one of the two ways:

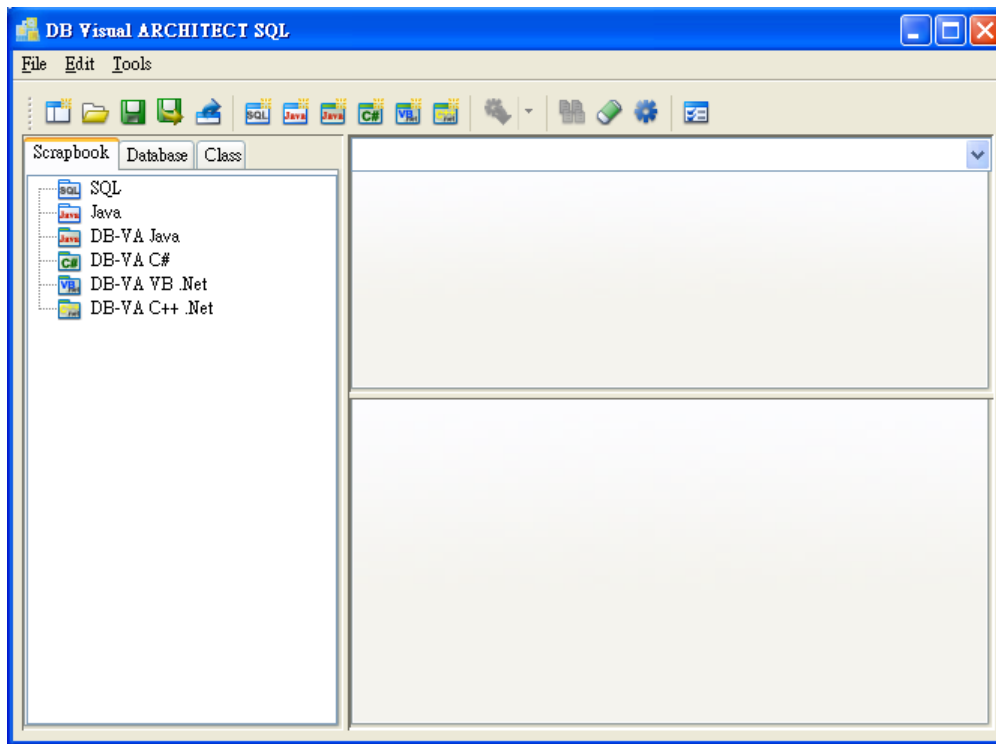
- On the Start Menu, select **Programs > Visual Paradigm > DB-VA SQL 1.0**, click **DB-VA SQL 1.0** shortcut to start the DB-VA SQL.



- Double-click the executable file named `run_dbvasql.exe` at the `<VP_Suite_Installation_Directory>\launcher` directory.



The DB-VA SQL is running in a separate window which is independent of the DB-VA.



Standalone version of DB-VA SQL can be launched independently. To run either VP-UML, DB-VA or SDE, and DB-VA SQL separately and simultaneously, different workspace must be selected for the two applications.

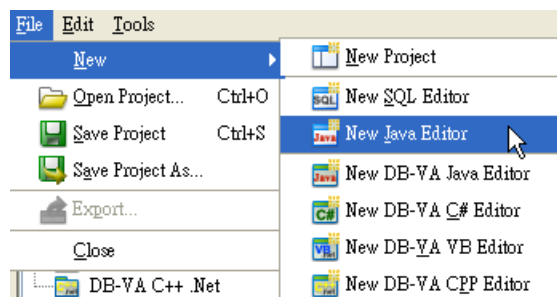
Using the Editor

DB Visual ARCHITECT SQL (DB-VA SQL) provides a text editor feature for editing the SQL statements, Java code, Java persistence code and .NET persistence code which are generated by either VP-UML, DB-VA or SDE. The text editor enables syntax highlight according to the type of source editor used.

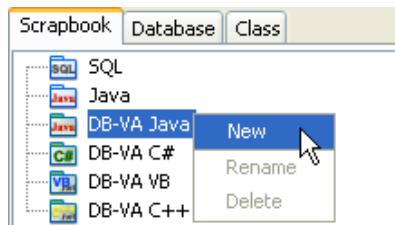
Creating a new Editor

A new editor can be added in one of the three ways:

- On the menu, click **File > New**, click one type of the editors.



- Right-click one type of the editors, select **New** from the pop-up menu.



- On the toolbar, click an icon for a particular type of the editors.



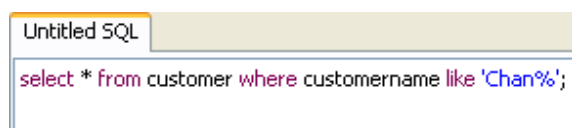
A new text editor is created. You can then edit the SQL statements, Java code, Java persistence code and .Net persistence code manually.

Editing on the Editor

As syntax highlight is enabled, typing mistake for the reserved words can be easily observed. Let's take the SQL editor and Java editor as examples.

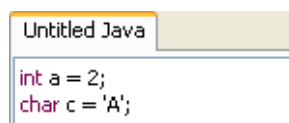
SQL Editor

As the select statement with specified condition has a standard format, "Select <columns> from <table> where <condition>", the reserved words are highlighted in red.



Java Editor

As the declaration statement of Java has a standard format, "data_type identifier = initial_value", the primitive data types are highlighted in red.

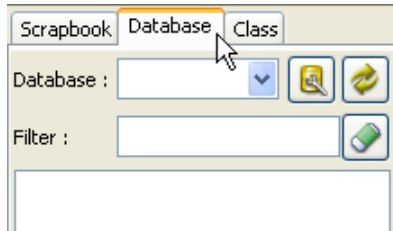


Configuring Database Connection

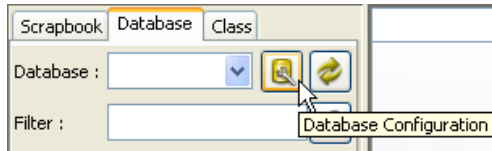
As DB Visual ARCHITECT SQL (DB-VA SQL) enables the real-time database manipulation, database must be connected to allow altering the database.

To configure database connection:

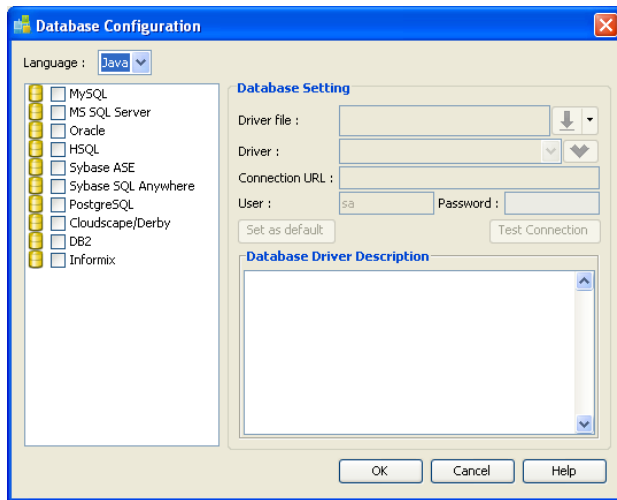
1. Click the **Database** tab pane.




2. Click **Database Configuration** button to display the **Database Configuration** dialog box.

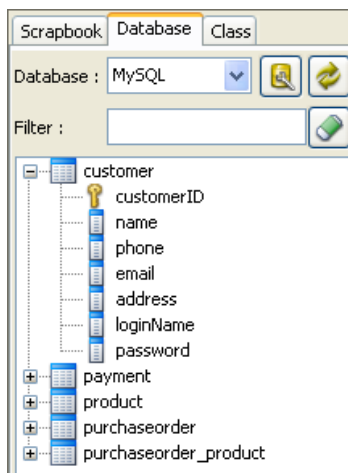


3. Configure the database connection by selecting the database and defining the database setting.



 Opening an existing project which has been created and configured with database in one of the VP's products, the connected database will be automatically loaded in to the DB-VA SQL environment.

After configured the database connection, the tables are listed automatically. By clicking on the plus sign (+), the columns of the table will be shown.



Generating SQL Statement and Java Code for Database Manipulation

Having connected to database, DB Visual ARCHITECT SQL (DB-VA SQL) is capable of generating SQL statements and Java

code to manipulate the selected database; i.e. retrieving, inserting, updating and deleting records.

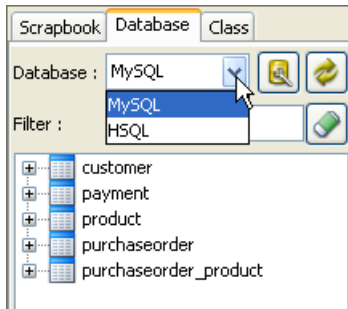
Generating SQL Statement

SQL statements are used to access and manipulate the database. Using the Select, Insert Into, Update and Delete statements, records can be retrieved, added, updated and deleted accordingly. Table can also be deleted from the database by using the Drop statement.

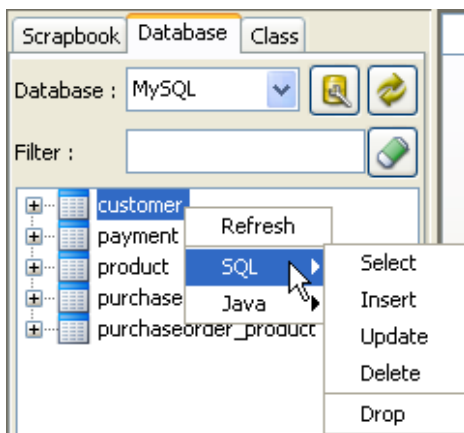
DB-VA SQL generates the SQL statements with a particular clause according to your selections.

To generate the SQL statement with a particular clause, perform the following steps:

1. Click the **Database** tab pane, select the connected database from the drop-down menu of **Database**.



2. Select the desired table(s) and/or column(s).
3. Right-click on a selected element, a pop-up menu is shown.
4. Select the desired clause for the generation of SQL statement from the sub-menu of **SQL**.



Generating SQL Statement upon Selection

SQL statements are generated according to your selections on the connected database. The selections can be identified into two types:

- Selecting from one table
- Selecting multiple tables

Examples are given to show the generated SQL statements on different type of selection.



DB-VA SQL generates a question mark (?) in the SQL statements indicating that it should be replaced by a value.



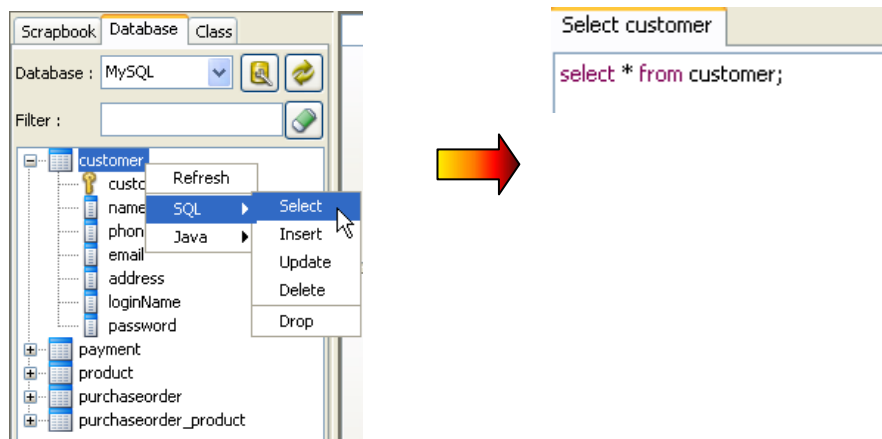
You are allowed to modify the generated SQL statements. To test the validity of the SQL statements, simply execute the SQL statements to check the result.

Selecting from One Table

The following examples show the generated SQL statements with respect to the selections on a particular table.

Examples:

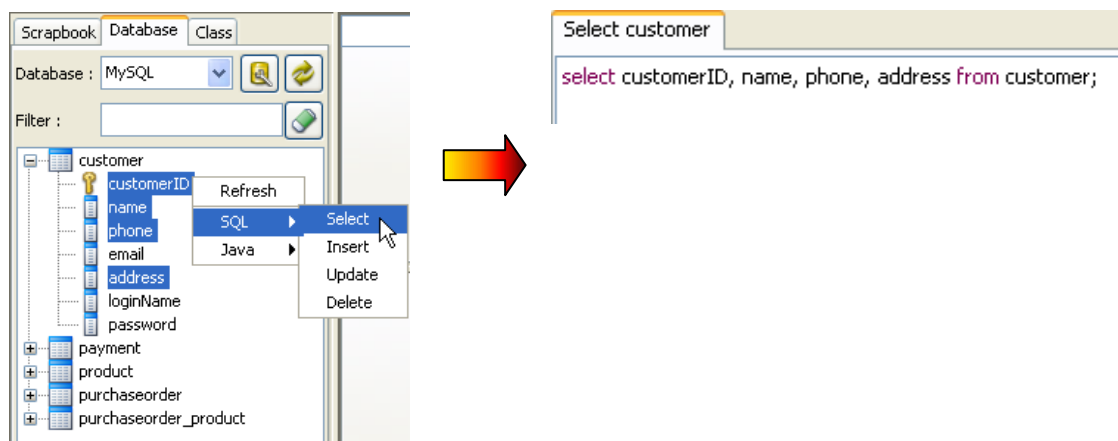
Generating Select SQL statement by selecting Customer table:



SQL statement is generated as follows:

```
select * from customer;
```

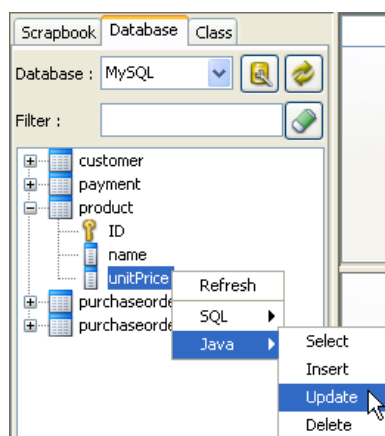
Generating Select SQL statement by selecting customer's ID, name, phone and address from Customer table:



SQL statement is generated as follows:

```
select customerID, name, phone, address from customer;
```

Generating Update SQL statement by selecting unit price from Product table:

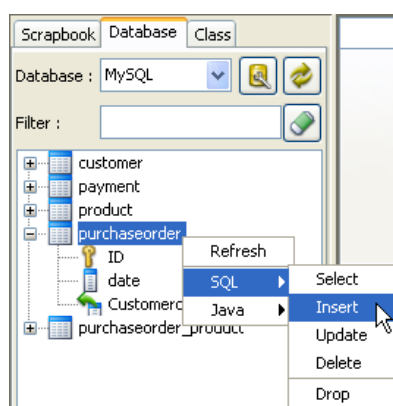


```
Update product
update product set unitPrice = ? where ID = ?;
```

SQL statement is generated as follows:

```
update product set unitPrice = ? where ID = ?;
```

Generating Insert SQL statement by selecting Purchase Order table:

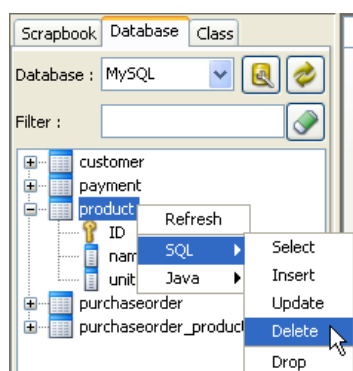


```
Insert purchaseorder
insert into purchaseorder (ID, date, CustomercustomerID) values (?, ?, ?);
```

SQL statement is generated as follows:

```
insert into purchaseorder (ID, date, CustomercustomerID) values (?, ?, ?);
```

Generating Delete SQL statement by selecting Product table:



```
Delete product
delete from product where ID = ?;
```

SQL statement is generated as follows:

```
delete from product where ID = ?;
```

Selecting from Multiple Tables

DB-VA SQL supports generating Select SQL statements on multiple selected tables by adding the Where-clause for the condition. The Where-clause is generated based on the foreign key constraints defined in the database. The following examples show the generated Select SQL statements by selecting multiple tables.



If the database does not support foreign key constraints, the Where-clause will not be generated.

Examples:

Generating Select SQL statement by selecting customer's name, phone and purchase order's ID and date from Customer and Purchase Order tables respectively:

The screenshot shows the 'customer' table selected in the 'SQL' menu. The generated SQL statement is:

```
Select customer, purchaseorder
select customer.name, customer.phone, purchaseorder.ID, purchaseorder.date
from customer, purchaseorder
where purchaseorder.CustomercustomerID = customer.customerID;
```

SQL statement is generated as follows:

```
select customer.name, customer.phone, purchaseorder.ID, purchaseorder.date
from customer, purchaseorder
where purchaseorder.CustomercustomerID = customer.customerID;
```

Generating Select SQL statement by selecting product's name, purchase order's id and date and purchaseorder_product's qty from Product, Purchase Order and PurchaseOrder_Prdouct tables accordingly:

The screenshot shows the 'purchaseorder' table selected in the 'SQL' menu. The generated SQL statement is:

```
Select purchaseorder, product, purchaseorder_product
select purchaseorder.ID, purchaseorder.date, product.name, purchaseorder_product.qty
from purchaseorder, product, purchaseorder_product
where purchaseorder_product.PurchaseOrderID = purchaseorder.ID
and purchaseorder_product.ProductID = product.ID;
```

SQL statement is generated as follows:

```
select purchaseorder.ID, purchaseorder.date, product.name, purchaseorder_product.qty
```

```

from purchaseorder, product, purchaseorder_product
where purchaseorder_product.PurchaseOrderID = purchaseorder.ID
and purchaseorder_product.ProductID = product.ID;

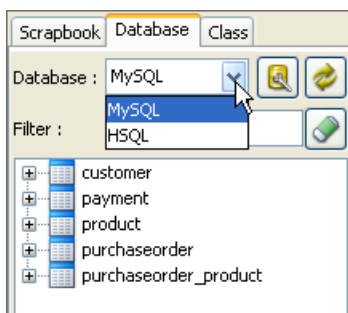
```

Generating Java Code

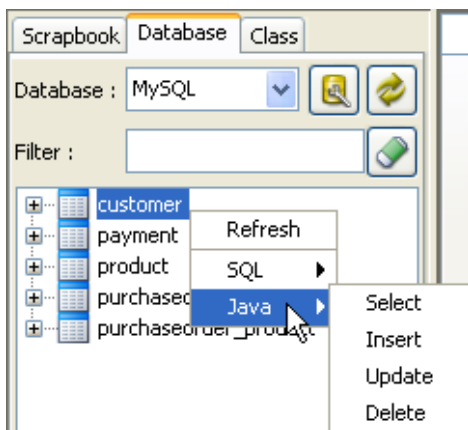
When developing database application with Java, Java code implementing with SQL statements must be used so as to access and manipulate the database. DB-VA SQL generates Java code to manipulate the database based on your selections; i.e. retrieving, inserting, updating and deleting records from database. You can thus apply the generated Java code to your implementation.

To generate the Java code for a particular action, perform the following steps:

1. Click the **Database** tab pane, select the connected database from the drop-down menu of **Database**.



2. Select the desired table(s) and/or column(s).
3. Right-click on a selected element, a pop-up menu is shown.
4. Select the desired type of manipulation for the generation of Java code from the sub-menu of **Java**.



Generating Java Code upon Selection

Java code is generated according to your selections on the connected database. The selections can be identified into two types:

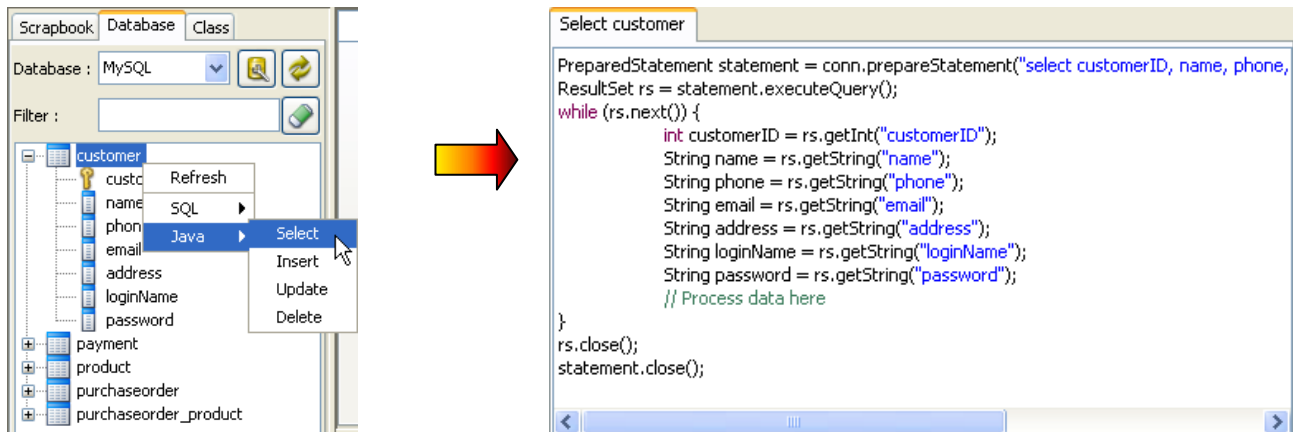
- Selecting from one table
- Selecting multiple tables

Examples are given to show the generated Java Code on different type of selection.

Selecting from One Table

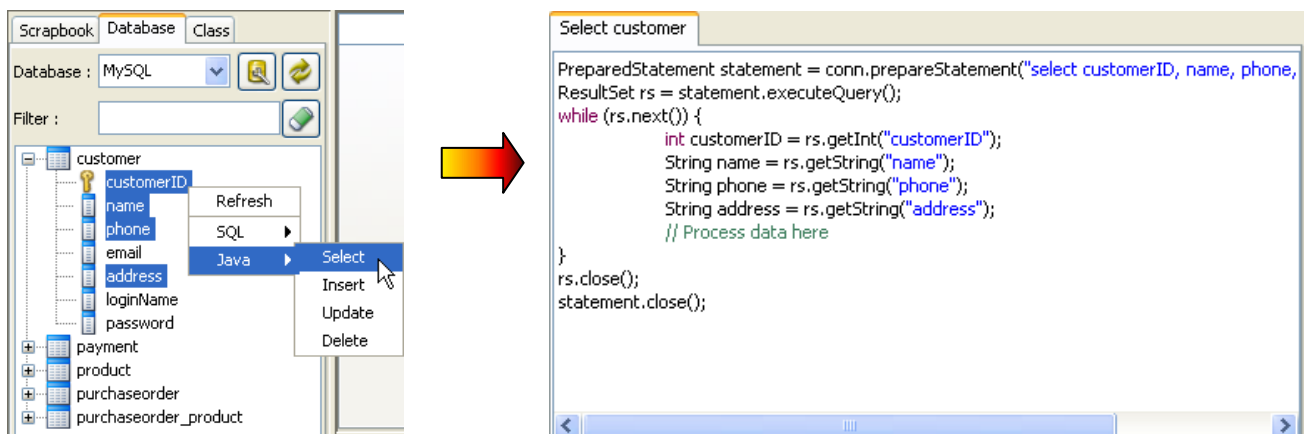
The following examples show the generated Java code with respect to the selections on a particular table.

Examples:

Generating Java code with Select SQL statement by selecting Customer table:

Java code is generated as follows:

```
PreparedStatement statement = conn.prepareStatement("select customerID,
    name, phone, email, address, loginName, password from customer");
ResultSet rs = statement.executeQuery();
while (rs.next()) {
    int customerID = rs.getInt("customerID");
    String name = rs.getString("name");
    String phone = rs.getString("phone");
    String email = rs.getString("email");
    String address = rs.getString("address");
    String loginName = rs.getString("loginName");
    String password = rs.getString("password");
    // Process data here
}
rs.close();
statement.close();
```

Generating Java code with Select SQL statement by selecting customer's ID, name, phone and address from Customer table:

Java code is generated as follows:

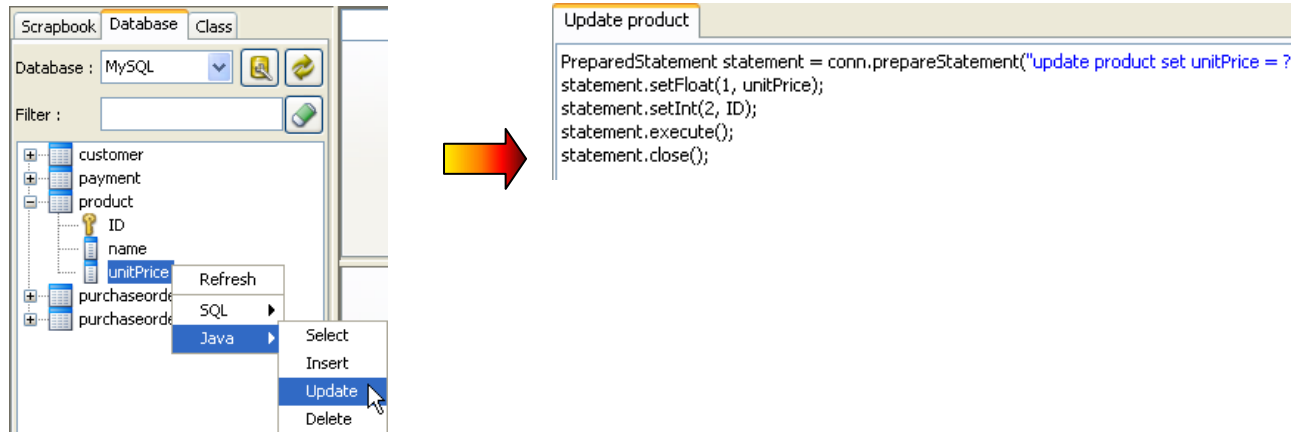
```
PreparedStatement statement = conn.prepareStatement("select customerID, name, phone,
    address from customer");
ResultSet rs = statement.executeQuery();
while (rs.next()) {
    int customerID = rs.getInt("customerID");
    String name = rs.getString("name");
    String phone = rs.getString("phone");
```

```

String address = rs.getString("address");
// Process data here
}
rs.close();
statement.close();

```

Generating Java code with Update SQL statement by selecting unit price from Product table:



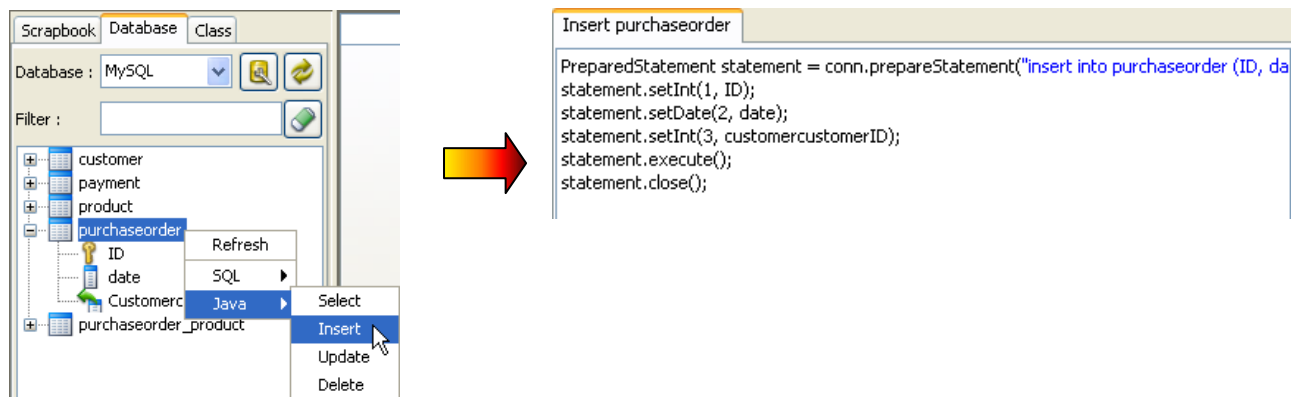
Java code is generated as follows:

```

PreparedStatement statement = conn.prepareStatement("update product
set unitPrice = ? where ID = ?");
statement.setFloat(1, unitPrice);
statement.setInt(2, ID);
statement.execute();
statement.close();

```

Generating Java code with Insert SQL statement by selecting Purchase Order table:



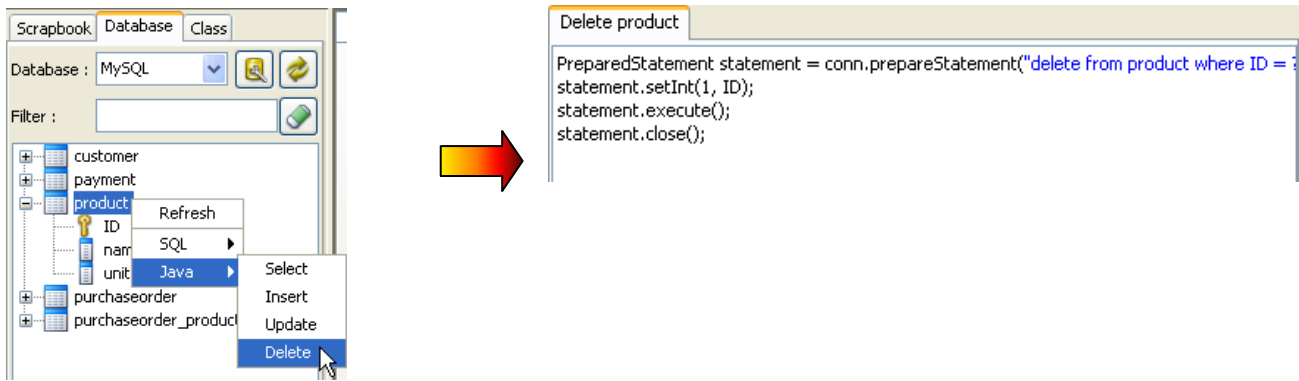
Java code is generated as follows:

```

PreparedStatement statement = conn.prepareStatement("insert into purchaseorder (ID, date,
CustomercustomerID) values (?, ?, ?)");
statement.setInt(1, ID);
statement.setDate(2, date);
statement.setInt(3, CustomercustomerID);
statement.execute();
statement.close();

```

Generating Java code with Delete SQL statement by selecting Product table:



Java code is generated as follows:

```
PreparedStatement statement = conn.prepareStatement("delete from product
    where ID = ?");
statement.setInt(1, ID);
statement.execute();
statement.close();
```

Selecting from Multiple Tables

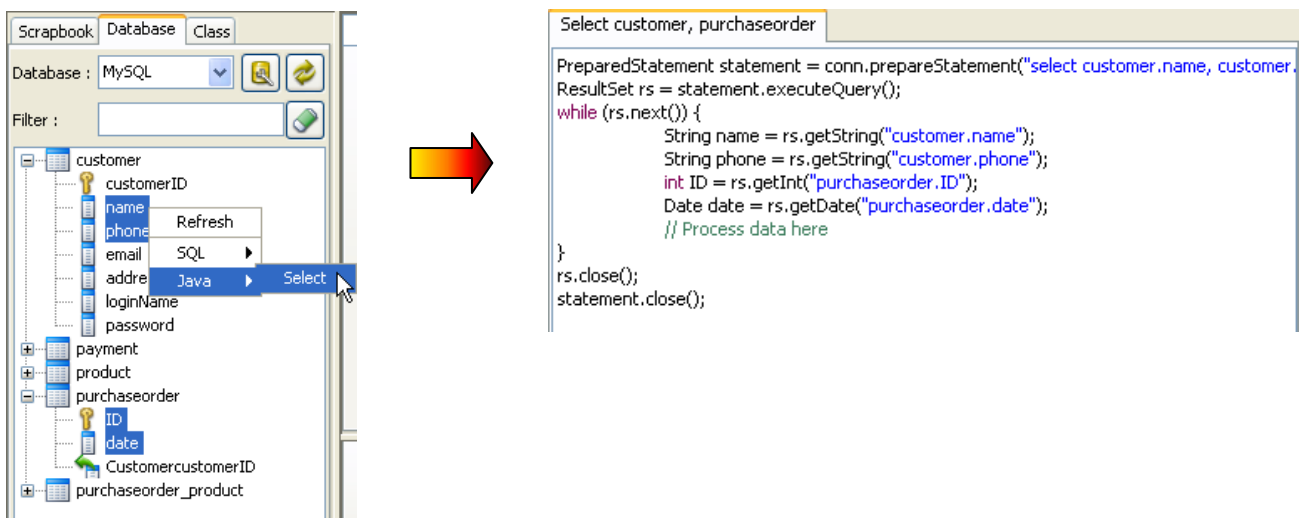
DB-VA SQL supports generating Java code with Select SQL statements with Where-clause for selecting on multiple tables. The Where-clause is generated based on the foreign key constraints defined in the database. The following examples show the generated Select SQL statements by selecting multiple tables.



If the database does not support foreign key constraints, the Where-clause will not be generated.

Examples:

Generating Java code with Select SQL statement by selecting customer's name, phone and purchase order's ID and date from Customer and Purchase Order tables respectively:



Java code is generated as follows:

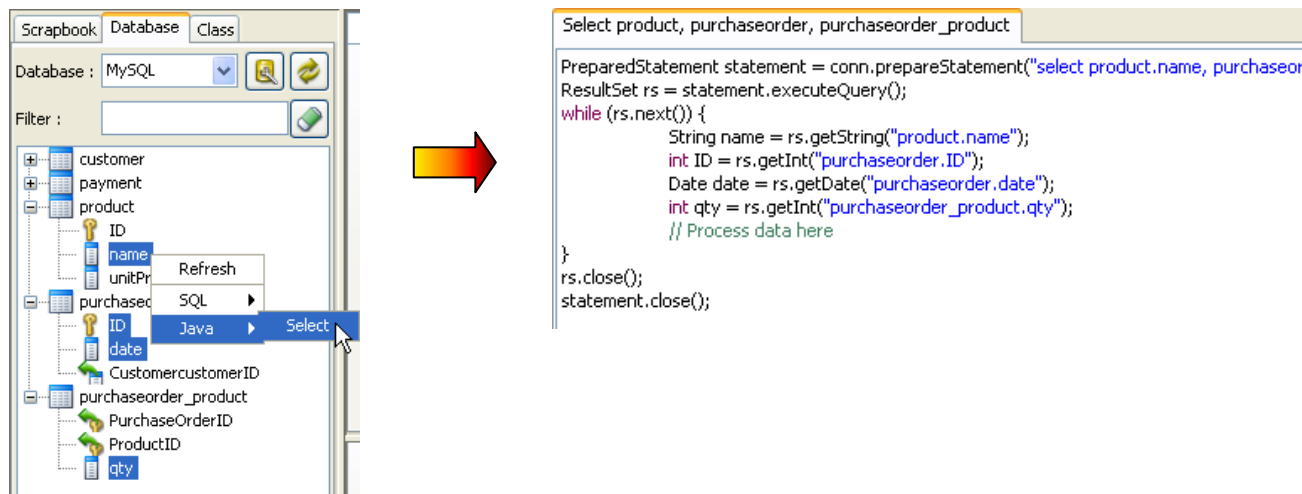
```
PreparedStatement statement = conn.prepareStatement("select customer.name,
    customer.phone, purchaseorder.ID, purchaseorder.date from customer,
    purchaseorder where purchaseorder.CustomercustomerID = customer.customerID");
ResultSet rs = statement.executeQuery();
```

```

while (rs.next()) {
    String name = rs.getString("customer.name");
    String phone = rs.getString("customer.phone");
    int ID = rs.getInt("purchaseorder.ID");
    Date date = rs.getDate("purchaseorder.date");
    // Process data here
}
rs.close();
statement.close();

```

Generating Java code with Select SQL statement by selecting product's name, purchase order's id and date and purchaseorder_product's qty from Product, Purchase Order and PurchaseOrder_Product tables accordingly:



Java code is generated as follows:

```

PreparedStatement statement = conn.prepareStatement("select product.name,
purchaseorder.ID, purchaseorder.date, purchaseorder_product.qty from product,
purchaseorder, purchaseorder_product where purchaseorder_product.PurchaseOrderID =
purchaseorder.ID and purchaseorder_product.ProductID = product.ID");
ResultSet rs = statement.executeQuery();
while (rs.next()) {
    String name = rs.getString("product.name");
    int ID = rs.getInt("purchaseorder.ID");
    Date date = rs.getDate("purchaseorder.date");
    int qty = rs.getInt("purchaseorder_product.qty");
    // Process data here
}
rs.close();
statement.close();

```

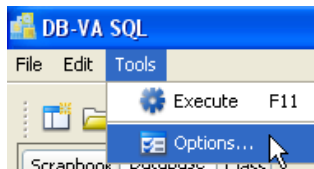
Modifying the Name of Database Connection Identifier

In order to access and alter the database, a connection with a specific database must be defined in the Java code. By default, the connection variable is "conn" representing the connection with the selected database. You are allowed to change the name of the connection variable.

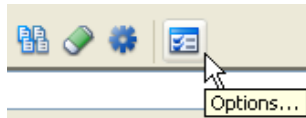
To change the name of the connection variable:

1. Activate the Options dialog box in one of the two ways:

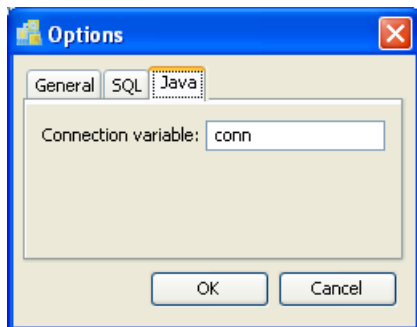
- On the menu, click **Tools > Options...**



- On the toolbar, click the **Options...** icon.



2. Click **Java** tab, and enter the desired name for the **Connection variable**.



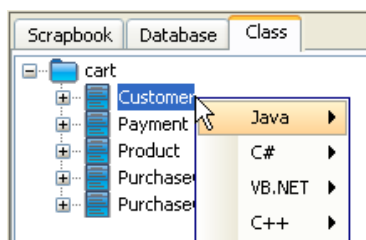
Generating Persistence Code for Database Manipulation

Having generated persistence classes, either Java or .NET with VP-UML, DB-VA or SDE, the persistence class provides methods to manipulate the database directly. DB Visual ARCHITECT SQL (DB-VA SQL) is able to generate the persistence code which is based on the methods generated in the persistence class for database manipulation. Hence, there is a prerequisite of generating persistence code for database manipulation; that is, the persistence classes must be generated by VP-UML, DB-VA or SDE in advance.

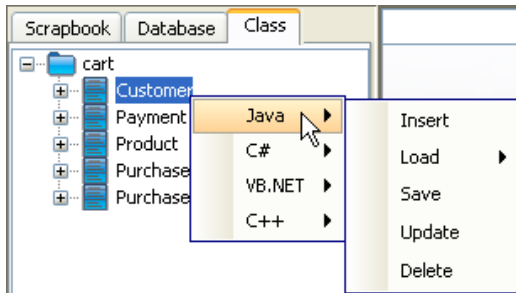
As there are four types of persistent API available for generating the persistence class, DB-VA SQL will generate the persistence code with respect to the selected persistent API. You can thus implement your project with the generated persistence class and apply the persistence code to manipulate the database.

To generate the persistence code for a particular action, perform the following steps:

1. Click the **Class** tab pane, select the
2. Select the desired class(s) and/or attribute(s).
3. Right-click on a selected element, a pop-up menu is shown.
4. Select the desired type of persistence code to be generated, either **Java**, **C#**, **VB.NET** or **C++**.



5. Select the desired type of manipulation from the sub-menu of the selected type of persistence code.

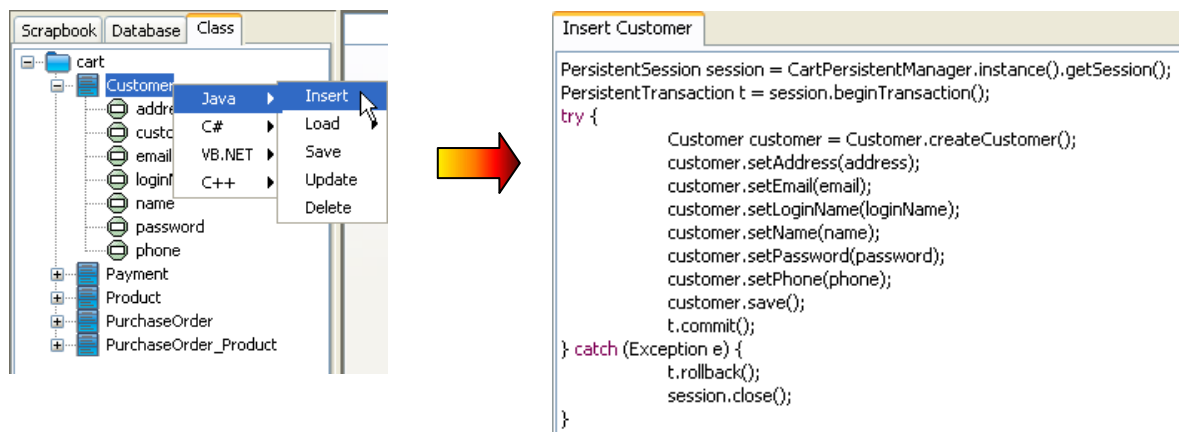


Generating Persistence Code for Inserting Record

The following examples show the generated persistence code with static method for inserting a new record to the database corresponding to the selected class.

Examples:

Generating Java persistence code for inserting a record to Customer table:



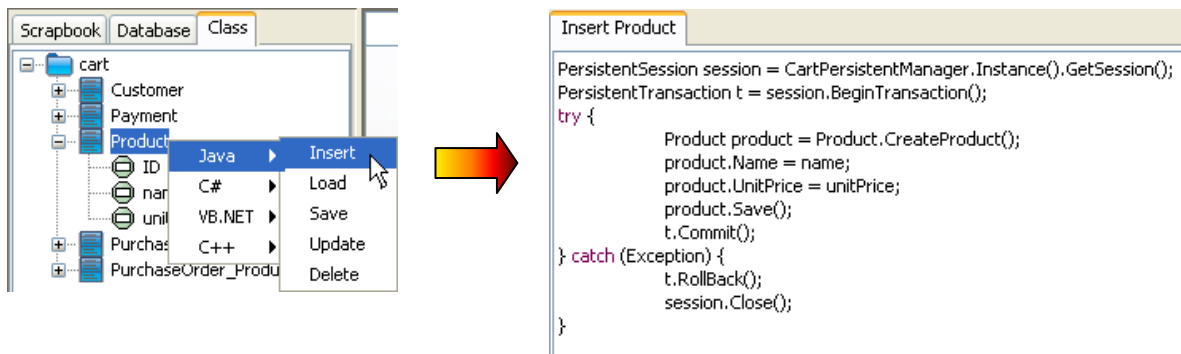
Java persistence code is generated as follows:

```

PersistentSession session = CartPersistentManager.instance().getSession();
PersistentTransaction t = session.beginTransaction();
try {
    Customer customer = Customer.createCustomer();
    customer.setAddress(address);
    customer.setEmail(email);
    customer.setLoginName(loginName);
    customer.setName(name);
    customer.setPassword(password);
    customer.setPhone(phone);
    customer.save();
    t.commit();
} catch (Exception e) {
    t.rollback();
    session.close();
}

```

Generating C# persistence code for inserting a record to Product table:



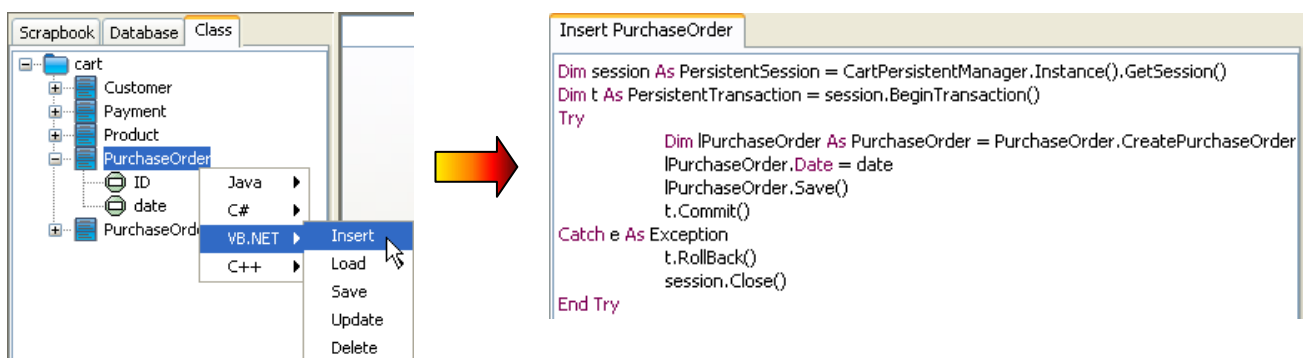
C# persistence code is generated as follows:

```

PersistentSession session = CartPersistentManager.Instance().GetSession();
PersistentTransaction t = session.BeginTransaction();
try {
    Product product = Product.CreateProduct();
    product.Name = name;
    product.UnitPrice = unitPrice;
    product.Save();
    t.Commit();
} catch (Exception) {
    t.Rollback();
    session.Close();
}

```

Generating VB.NET persistence code for inserting a record to Purchase Order table:



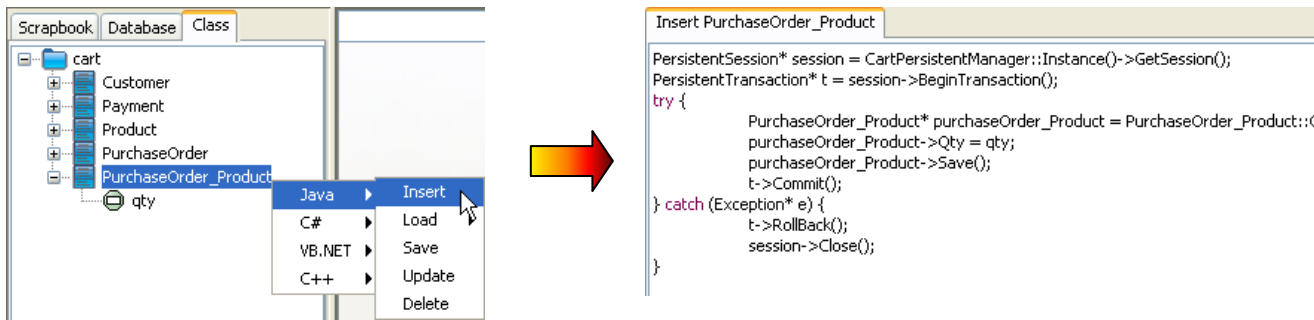
VB.NET persistence code is generated as follows:

```

Dim session As PersistentSession = CartPersistentManager.Instance().GetSession()
Dim t As PersistentTransaction = session.BeginTransaction()
Try
    Dim lPurchaseOrder As PurchaseOrder = PurchaseOrder.CreatePurchaseOrder()
    lPurchaseOrder.Date = date
    lPurchaseOrder.Save()
    t.Commit()
Catch e As Exception
    t.Rollback()
    session.Close()
End Try

```

Generating C++ persistence code for inserting a record to PurchaseOrder_Product table:



C++ persistence code is generated as follows:

```
PersistentSession* session = CartPersistentManager::Instance()->GetSession();
PersistentTransaction* t = session->BeginTransaction();
try {
    PurchaseOrder_Product* purchaseOrder_Product =
        PurchaseOrder_Product::CreatePurchaseOrder_Product();
    purchaseOrder_Product->Qty = qty;
    purchaseOrder_Product->Save();
    t->Commit();
} catch (Exception* e) {
    t->RollBack();
    session->Close();
}
```

Generating Persistence Code for Retrieving a Record

The persistence class is generated with the load methods which support retrieving a record from database. There are three types of load methods generated:

- By primary key
- By Query
- By ORM Qualifier

By default, the load by primary key and query methods are generated which allows the record retrieval by specifying the primary key and user defined condition respectively. The load by ORM Qualifier method will be generated if the extra data retrieval rules are defined in ORM Qualifiers.

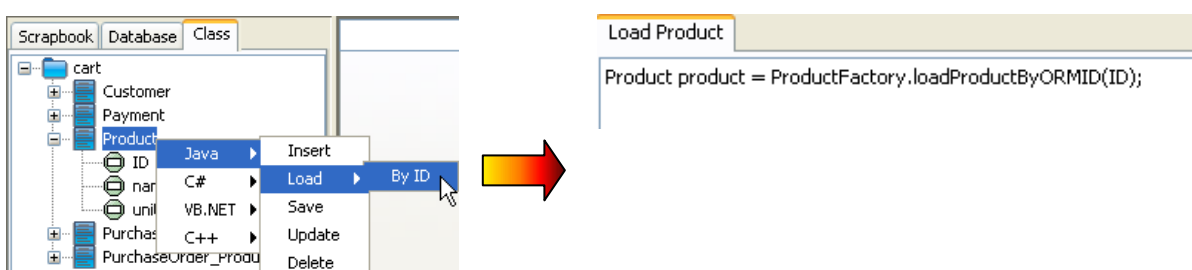
DB-VA SQL generates persistence code to retrieve record according to the type of load method selected. Examples are given to show the generated persistence code on different type of load methods selected.

Load by Primary Key

The following examples show the generated persistence code with factory class for retrieving a record by the primary key.

Examples:

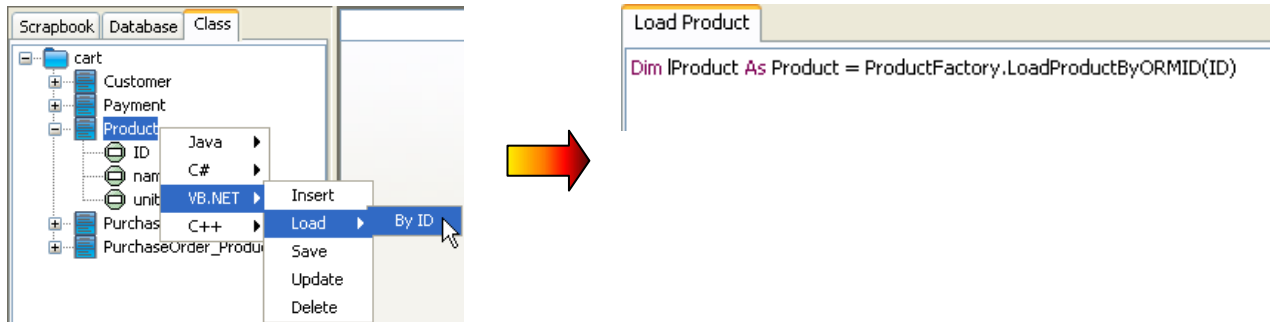
Generating Java persistence code for retrieving a Product record by specifying the primary key:



Java persistence code is generated as follows:

```
Product product = ProductFactory.loadProductByORMID(ID);
```

Generating VB.NET persistence code for retrieving a Product record by specifying the primary key:



```
Dim IProduct As Product = ProductFactory.LoadProductByORMID(ID)
```

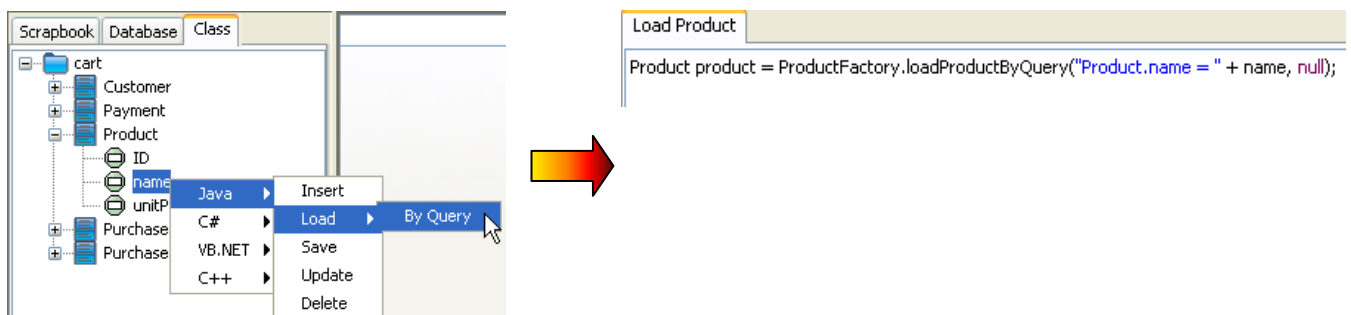
VB.NET persistence code is generated as follows:

```
Dim IProduct As Product = ProductFactory.LoadProductByORMID(ID)
```

Load by Query

The following examples show the generated persistence code with factory class for retrieving a record by user defined condition.

Generating Java persistence code for retrieving a Product record by specifying the product name:

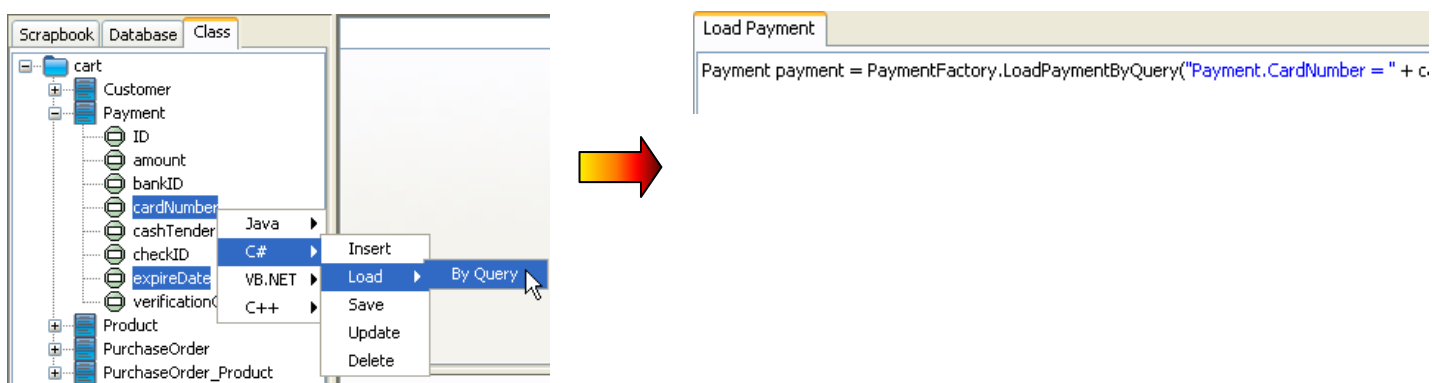


```
Product product = ProductFactory.loadProductByQuery("Product.name = " + name, null);
```

Java persistence code is generated as follows:

```
Product product = ProductFactory.loadProductByQuery("Product.name = " + name, null);
```

Generating C# persistence code for retrieving a Payment record by specifying the card number and expiry date:



```
Payment payment = PaymentFactory.LoadPaymentByQuery("Payment.CardNumber = " + c
```

C# persistence code is generated as follows:

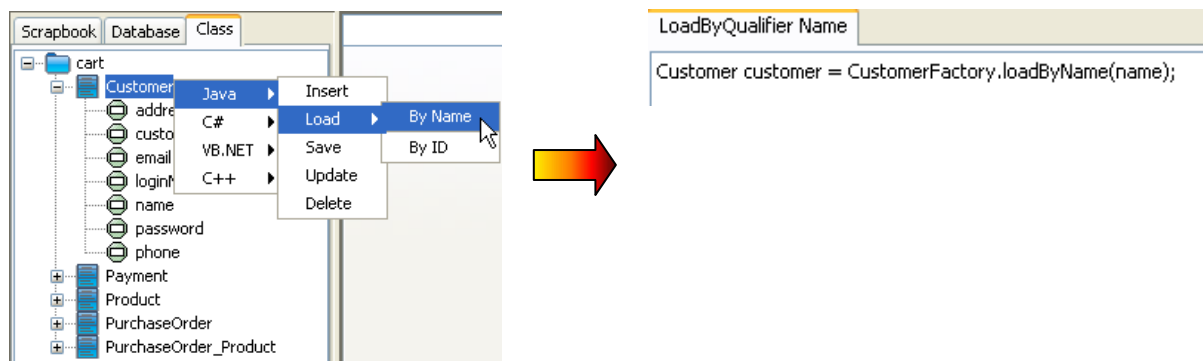
```
Payment payment = PaymentFactory.LoadPaymentByQuery("Payment.CardNumber = " + cardNumber + " and Payment.ExpireDate = " + expireDate, null);
```

Load by ORM Qualifier

The following examples show the generated persistence code with factory class for retrieving a record by the defined ORM Qualifier.

Examples:

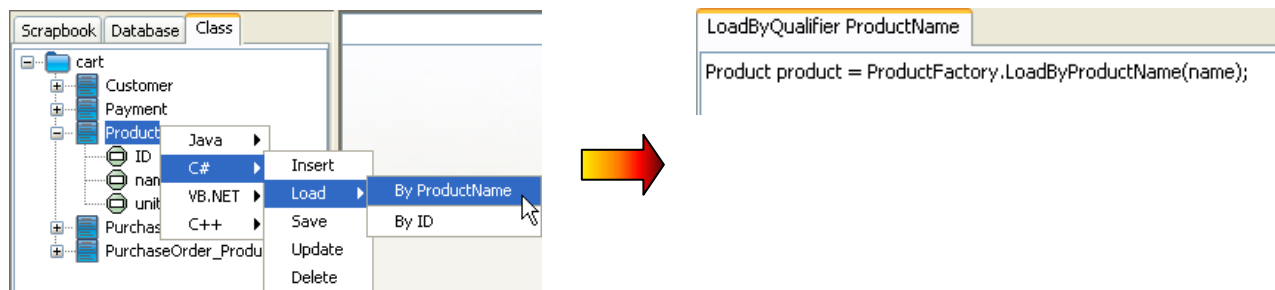
Generating Java persistence code for retrieving a Customer record by the ORM Qualifier of Name:



Java persistence code is generated as follows:

```
Customer customer = CustomerFactory.loadByName(name);
```

Generating C# persistence code for retrieving a Product record by the ORM Qualifier of ProductName:



C# persistence code is generated as follows:

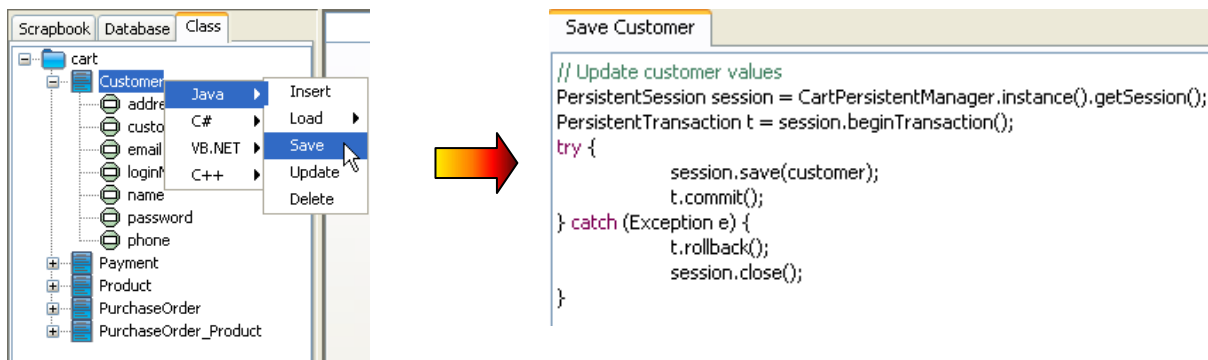
```
Product product = ProductFactory.LoadByProductName(name);
```

Generating Persistence Code for Saving a Record

The following examples show the generated persistence code with POJO for saving a record to the database.

Examples:

Generating Java persistence code for saving a Customer record:



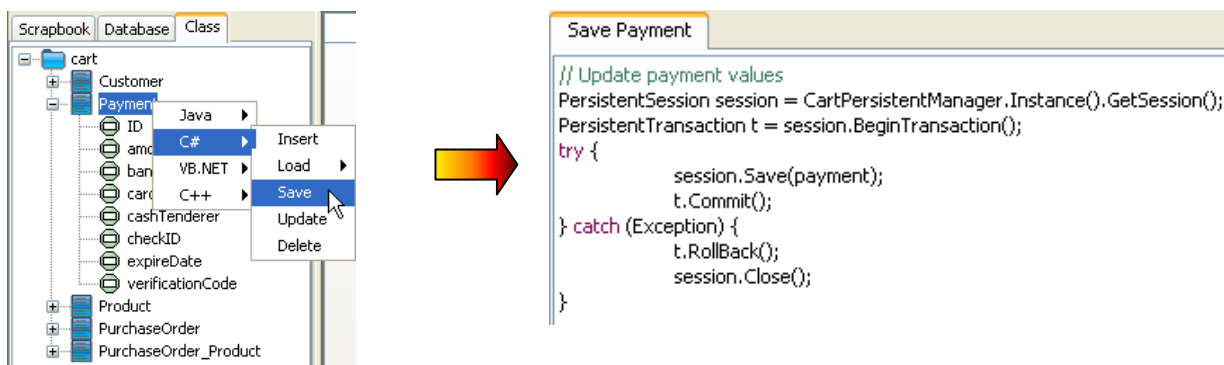
Java persistence code is generated as follows:

```

// Update customer values
PersistentSession session = CartPersistentManager.instance().getSession();
PersistentTransaction t = session.beginTransaction();
try {
    session.save(customer);
    t.commit();
} catch (Exception e) {
    t.rollback();
    session.close();
}

```

Generating C# persistence code for saving a Payment record:



C# persistence code is generated as follows:

```

// Update payment values
PersistentSession session = CartPersistentManager.Instance().GetSession();
PersistentTransaction t = session.BeginTransaction();
try {
    session.Save(payment);
    t.Commit();
} catch (Exception) {
    t.Rollback();
    session.Close();
}

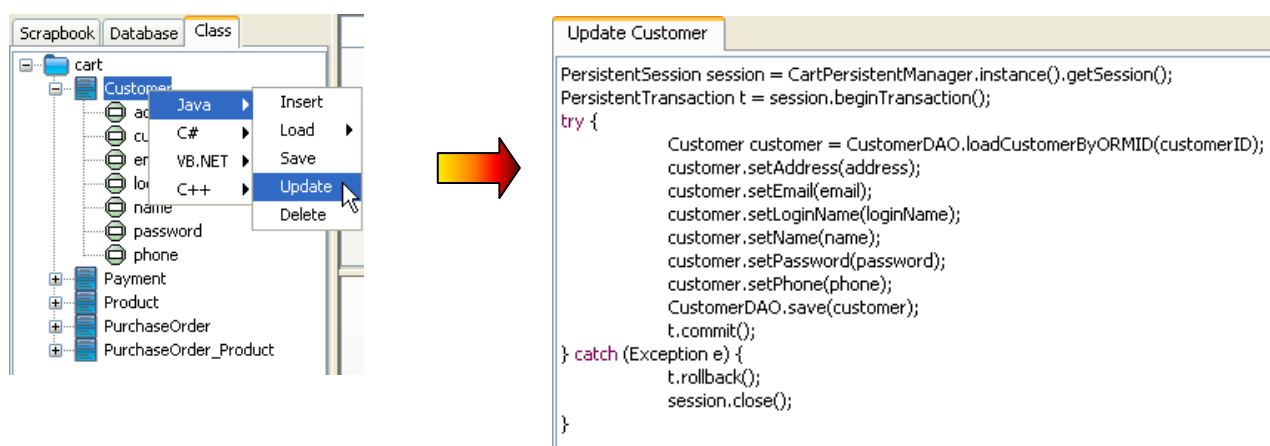
```

Generating Persistence Code for Updating a Record

The following examples show the generated persistence code with DAO for updating a record to the database corresponding to the selected class.

Examples:

Generating Java persistence code for updating a Customer record:



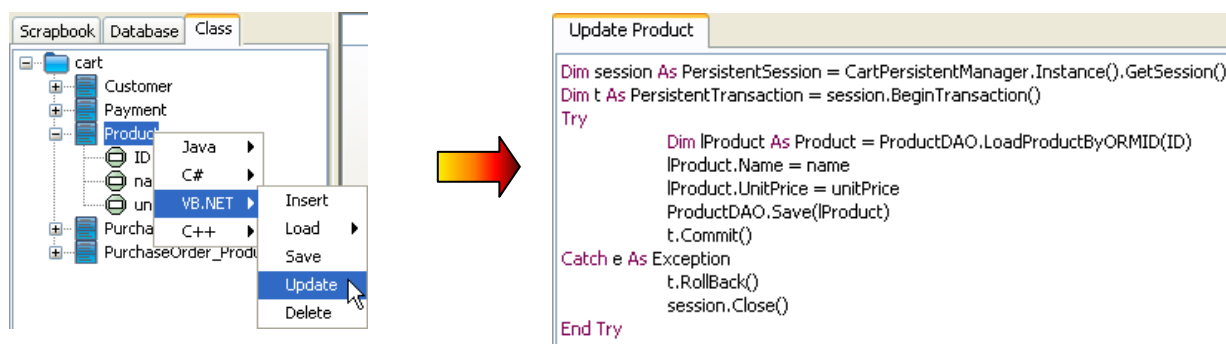
Java persistence code is generated as follows:

```

PersistentSession session = CartPersistentManager.instance().getSession();
PersistentTransaction t = session.beginTransaction();
try {
    Customer customer = CustomerDAO.loadCustomerByORMID(customerID);
    customer.setAddress(address);
    customer.setEmail(email);
    customer.setLoginName(loginName);
    customer.setName(name);
    customer.setPassword(password);
    customer.setPhone(phone);
    CustomerDAO.save(customer);
    t.commit();
} catch (Exception e) {
    t.rollback();
    session.close();
}

```

Generating VB.NET persistence code for updating a Product record:



VB.NET persistence code is generated as follows:

```

Dim session As PersistentSession = CartPersistentManager.Instance().GetSession()
Dim t As PersistentTransaction = session.BeginTransaction()
Try
    Dim lProduct As Product = ProductDAO.LoadProductByORMID(ID)
    lProduct.Name = name
    lProduct.UnitPrice = unitPrice
    ProductDAO.Save(lProduct)
    t.Commit()
Catch e As Exception
    t.Rollback()
    session.Close()
End Try

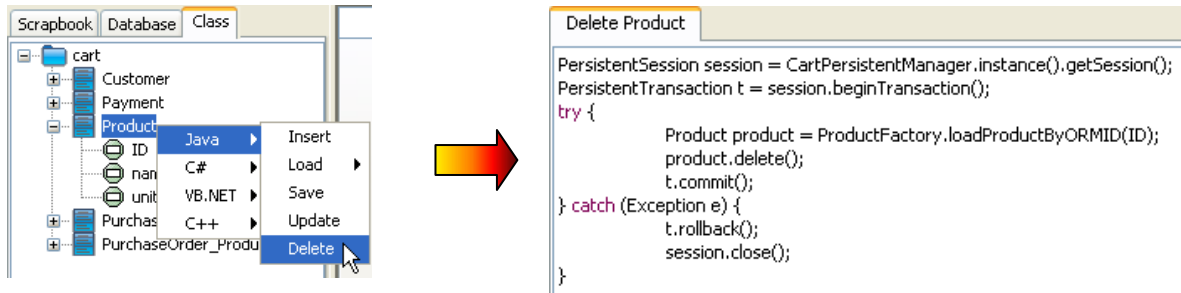
```

Generating Persistence Code for Deleting Record

The following examples show the generated persistence code with factory class for deleting a record from the database.

Examples:

Generating Java persistence code for deleting a Product record:



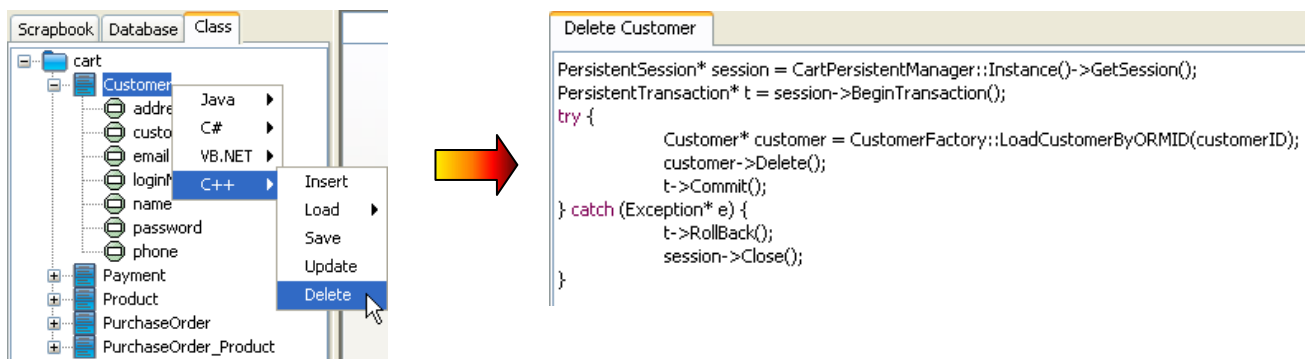
Java persistence code is generated as follows:

```

PersistentSession session = CartPersistentManager.instance().getSession();
PersistentTransaction t = session.beginTransaction();
try {
    Product product = ProductFactory.loadProductByORMID(ID);
    product.delete();
    t.commit();
} catch (Exception e) {
    t.rollback();
    session.close();
}

```

Generating C++ persistence code for deleting a Customer Record:



C++ persistence code is generated as follows:

```

PersistentSession* session = CartPersistentManager::Instance()->GetSession();
PersistentTransaction* t = session->BeginTransaction();
try {
    Customer* customer = CustomerFactory::LoadCustomerByORMID(customerID);
    customer->Delete();
    t->Commit();
} catch (Exception* e) {
    t->RollBack();
    session->Close();
}

```

Executing SQL Statements and Code

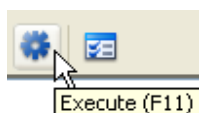
DB Visual ARCHITECT SQL (DB-VA SQL) allows you to execute the generated and/or updated SQL statements and code to manipulate the database in real-time and to test the modified statements and code. The result will show immediately.

You can execute the SQL statements and/or code by one of the three ways:

- On the menu, click **Tools > Execute**.



- On the toolbar, click the **Execute** icon.



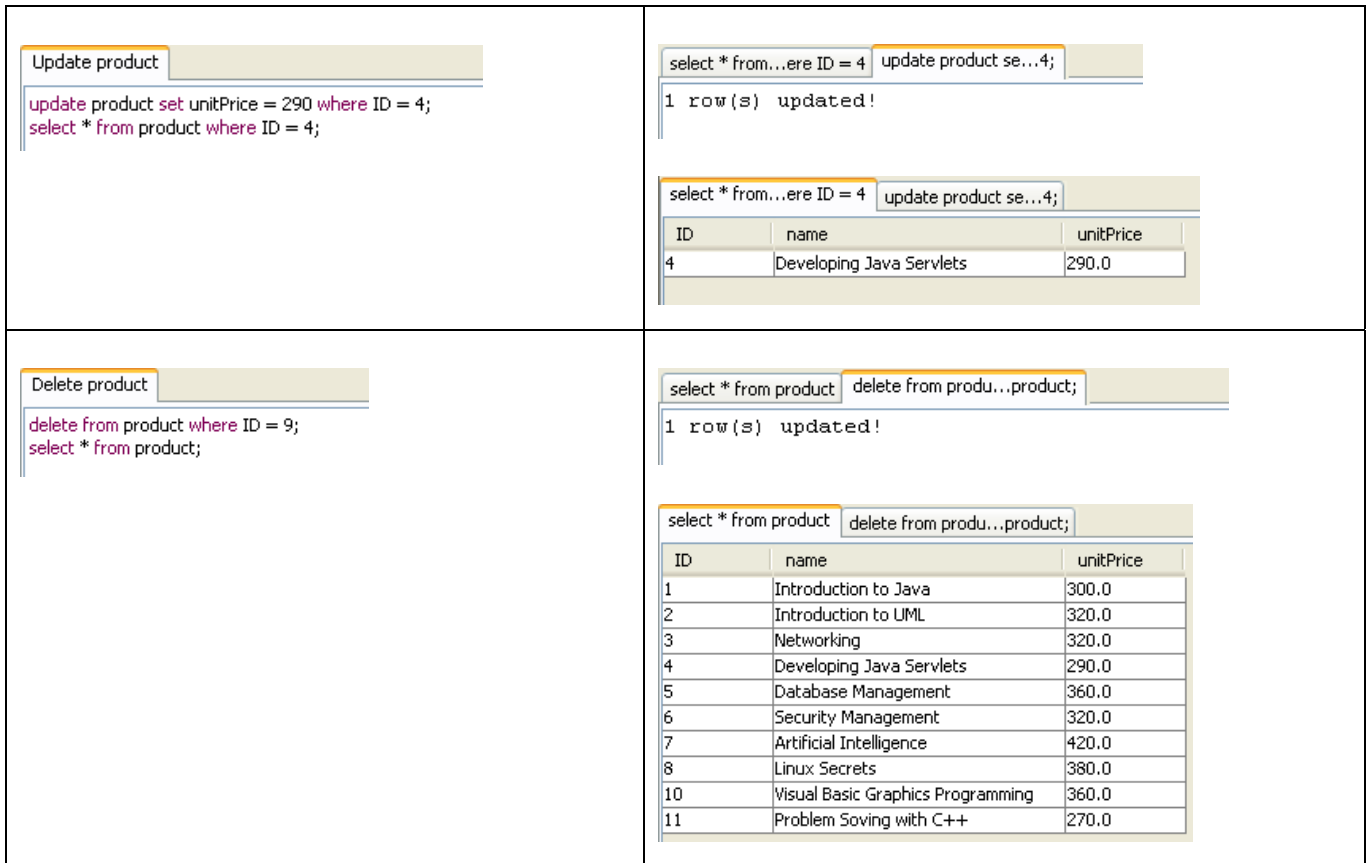
- Press **F11** button on the keyboard.

Executing SQL Statements

The following table shows the results of executing the SQL Statements.

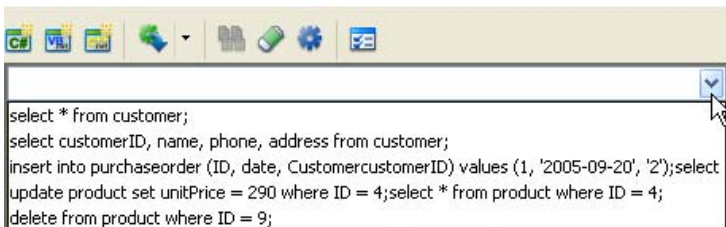
Examples:

SQL Statement	Result																														
<p>Select customer</p> <pre>select * from customer;</pre>	<p>select * from customer</p> <table border="1"> <thead> <tr> <th>customerID</th> <th>name</th> <th>phone</th> <th>email</th> <th>address</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Peter Pan</td> <td>23455678</td> <td>peter.pan@gmail.com</td> <td>123 Happy Valley</td> </tr> <tr> <td>2</td> <td>Shirley Kwong</td> <td>62457811</td> <td>shirley@yahoo.com</td> <td>3B, Ocean Court</td> </tr> <tr> <td>3</td> <td>Brian Wo</td> <td>23567815</td> <td>b_wo@yahoo.com.hk</td> <td>2/F, Garden Palace</td> </tr> <tr> <td>4</td> <td>Lydia Man</td> <td>68741259</td> <td>lydia_man@gmail.com</td> <td>7H, Irene Court</td> </tr> <tr> <td>5</td> <td>Ryan Chong</td> <td>91254387</td> <td>chong_r@gmail.com</td> <td>6B, Oit Building</td> </tr> </tbody> </table>	customerID	name	phone	email	address	1	Peter Pan	23455678	peter.pan@gmail.com	123 Happy Valley	2	Shirley Kwong	62457811	shirley@yahoo.com	3B, Ocean Court	3	Brian Wo	23567815	b_wo@yahoo.com.hk	2/F, Garden Palace	4	Lydia Man	68741259	lydia_man@gmail.com	7H, Irene Court	5	Ryan Chong	91254387	chong_r@gmail.com	6B, Oit Building
customerID	name	phone	email	address																											
1	Peter Pan	23455678	peter.pan@gmail.com	123 Happy Valley																											
2	Shirley Kwong	62457811	shirley@yahoo.com	3B, Ocean Court																											
3	Brian Wo	23567815	b_wo@yahoo.com.hk	2/F, Garden Palace																											
4	Lydia Man	68741259	lydia_man@gmail.com	7H, Irene Court																											
5	Ryan Chong	91254387	chong_r@gmail.com	6B, Oit Building																											
<p>Select customer</p> <pre>select customerID, name, phone, address from customer;</pre>	<p>select customerID, name, phone, address from customer</p> <table border="1"> <thead> <tr> <th>customerID</th> <th>name</th> <th>phone</th> <th>address</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Peter Pan</td> <td>23455678</td> <td>123 Happy Valley</td> </tr> <tr> <td>2</td> <td>Shirley Kwong</td> <td>62457811</td> <td>3B, Ocean Court</td> </tr> <tr> <td>3</td> <td>Brian Wo</td> <td>23567815</td> <td>2/F, Garden Palace</td> </tr> <tr> <td>4</td> <td>Lydia Man</td> <td>68741259</td> <td>7H, Irene Court</td> </tr> <tr> <td>5</td> <td>Ryan Chong</td> <td>91254387</td> <td>6B, Oit Building, 25 Oi Tai Street</td> </tr> </tbody> </table>	customerID	name	phone	address	1	Peter Pan	23455678	123 Happy Valley	2	Shirley Kwong	62457811	3B, Ocean Court	3	Brian Wo	23567815	2/F, Garden Palace	4	Lydia Man	68741259	7H, Irene Court	5	Ryan Chong	91254387	6B, Oit Building, 25 Oi Tai Street						
customerID	name	phone	address																												
1	Peter Pan	23455678	123 Happy Valley																												
2	Shirley Kwong	62457811	3B, Ocean Court																												
3	Brian Wo	23567815	2/F, Garden Palace																												
4	Lydia Man	68741259	7H, Irene Court																												
5	Ryan Chong	91254387	6B, Oit Building, 25 Oi Tai Street																												
<p>Insert purchaseorder</p> <pre>insert into purchaseorder (ID, date, CustomercustomerID) values (1, '2005-09-20', '2'); select * from purchaseorder;</pre>	<p>select * from purchaseorder insert into purchaseorder;</p> <p>1 row(s) updated!</p> <p>select * from purchaseorder insert into purchaseorder;</p> <table border="1"> <thead> <tr> <th>ID</th> <th>date</th> <th>CustomercustomerID</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>2005-09-20</td> <td>2</td> </tr> </tbody> </table>	ID	date	CustomercustomerID	1	2005-09-20	2																								
ID	date	CustomercustomerID																													
1	2005-09-20	2																													

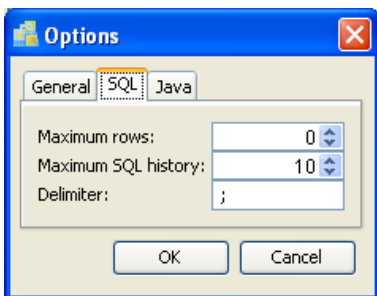


Showing Execution History

The recent execution on the SQL statements can be found from the drop-down menu under the toolbar. By selecting one of the SQL statements from history, a new SQL text editor will be generated with the selected SQL statement automatically; and hence, you can execute the SQL statement again.



By default, 10 of the recently executed SQL statements will be kept recorded in history. You can change the number of execution history stored by specifying the maximum number of execution on the SQL tab from the Options dialog box.

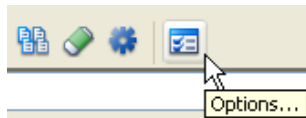


To change the number of execution history:

1. Activate the Options dialog box in one of the two ways:
 - On the menu, click **Tools > Options....**



- On the toolbar, click the **Options...** icon.



2. Click **SQL** tab, and enter the desired number of execution history to the **Maximum SQL history** field.

Executing Java Code

DB-VA SQL allows you to get the result of executing the Java code in real time by using the `println()` method. The `println()` method is the short form of the system output used in Java; i.e. `System.out.println()`. It provides a convenient way to test the Java code to be executed.

Taking the following generated Java code as an example,

```
Select customer
PreparedStatement statement = conn.prepareStatement("select customerID, name, phone,
ResultSet rs = statement.executeQuery();
while (rs.next()) {
    int customerID = rs.getInt("customerID");
    String name = rs.getString("name");
    String phone = rs.getString("phone");
    String email = rs.getString("email");
    String address = rs.getString("address");
    String loginName = rs.getString("loginName");
    String password = rs.getString("password");
    // Process data here
}
rs.close();
statement.close();
```

The `println()` method can be used after the comment to check the result. For example adding the following line of code after the comment:

```
println("Customer " + customerID + "/" + name);
```

After executing the Java code, the name for each customer will be displayed.

```
Select customer
Customer 1/ Peter Pan
Customer 2/ Shirley Kwong
Customer 3/ Brian Wo
Customer 4/ Lydia Man
Customer 5/ Ryan Chong

*** Terminated, elapsed time: 31 ms ***
```



To display the result of Java code execution, both `System.out.println()` and `println()` syntax are supported.

The following table shows the results of executing the Java code.

Examples:

Java Code	Result
-----------	--------

<pre> Select customer PreparedStatement statement = conn.prepareStatement("select customerID, name, phone, email, address, loginName, password from customer"); ResultSet rs = statement.executeQuery(); while (rs.next()) { int customerID = rs.getInt("customerID"); String name = rs.getString("name"); String phone = rs.getString("phone"); String email = rs.getString("email"); String address = rs.getString("address"); String loginName = rs.getString("loginName"); String password = rs.getString("password"); // Process data here println("Customer " + customerID + "/" + name); } rs.close(); statement.close(); </pre>	<pre> Select customer Customer 1/ Peter Pan Customer 2/ Shirley Kwong Customer 3/ Brian Wo Customer 4/ Lydia Man Customer 5/ Ryan Chong *** Terminated, elapsed time: 31 ms *** </pre>
<pre> Select customer PreparedStatement statement = conn.prepareStatement("select customerID, name, phone, address from customer"); ResultSet rs = statement.executeQuery(); while (rs.next()) { int customerID = rs.getInt("customerID"); String name = rs.getString("name"); String phone = rs.getString("phone"); String address = rs.getString("address"); // Process data here println("Customer ID: " + customerID); println("Name: " + name); println("Phone: " + phone); println("Address" + address); println("-----"); } rs.close(); statement.close(); </pre>	<pre> Select customer Customer ID: 1 Name: Peter Pan Phone: 23455678 Address123 Happy Valley ----- Customer ID: 2 Name: Shirley Kwong Phone: 62457811 Address3B, Ocean Court ----- Customer ID: 3 Name: Brian Wo </pre>
<pre> Update product float unitPrice = 290; int ID= 4; PreparedStatement statement = conn.prepareStatement("update product set unitPrice = ? where ID = ?"); statement.setFloat(1, unitPrice); statement.setInt(2, ID); statement.execute(); statement.close(); </pre>	<pre> Update product *** Terminated, elapsed time: 31 ms *** </pre>
<pre> Insert purchaseorder PreparedStatement statement = conn.prepareStatement("insert into purchaseorder (ID, date, CustomercustomerID) values (?, ?, ?)"); statement.setInt(1, 3); statement.setDate(2, new Date(2005, 9, 20)); statement.setInt(3, 3); statement.execute(); statement.close(); </pre>	<pre> Insert purchaseorder *** Terminated, elapsed time: 141 ms *** </pre>
<pre> Delete product int ID = 9; PreparedStatement statement = conn.prepareStatement("delete from product where ID = ?"); statement.setInt(1, ID); statement.execute(); statement.close(); </pre>	<pre> Delete product *** Terminated, elapsed time: 31 ms *** </pre>

Executing Persistence Code

DB-VA SQL allows you to execute the Java persistence code to check the result in real time. In order to execute the java persistence code, the persistence class must be compiled beforehand and stored in `$(output folder)\classes` folder. No execution result can be found if the compiled persistence classes are not stored in the proper file directory.

 DB-VA SQL only supports the execution of SQL statements, Java code and Java persistence code. .NET persistence code cannot be executed in DB-VA SQL.

In order to check the execution result of Java persistence code, you can simply use either the `println()` method or `System.out.println()` method to display the result.

Let us take the following generated Java persistence code as an example,

```
Load Product
Product product = ProductFactory.loadProductByORMID(ID);
```

The `println()` method can be used to display the retrieved product record. For example adding the following lines of code after the load by method.

```
println("Product Name " + product.getName());
println("Unit Price: " + product.getUnitPrice());
```

After executing the Java persistence code, the product's name and unit price will be displayed.

```
Load Product
Product Name: Introduction to UML
Unit Price: 320.0

*** Terminated, elapsed time: 297 ms ***
```

The following table shows the results of executing the Java persistence code.

Examples:

Java Persistence Code	Result
<pre>Insert Customer PersistentSession session = CartPersistentManager.instance().getSession(); PersistentTransaction t = session.beginTransaction(); try { Customer customer = Customer.createCustomer(); customer.setAddress("3A, Beautiful Garden"); customer.setEmail("steven.black@yahoo.com.us"); customer.setLoginName("steven"); customer.setName("Steven Black"); customer.setPassword("sblack"); customer.setPhone("85412367"); customer.save(); t.commit(); } catch (Exception e) { t.rollback(); session.close(); }</pre>	<pre>Insert Customer *** Terminated, elapsed time: 515 ms ***</pre>
<pre>Load Product Product product = ProductFactory.loadProductByORMID(2); println("Product Name : " + product.getName()); println("Unit Price: " + product.getUnitPrice());</pre>	<pre>Load Product Product Name: Introduction to UML Unit Price: 320.0 *** Terminated, elapsed time: 297 ms ***</pre>
<pre>Load Product String name = "Networking"; Product product = ProductFactory.loadProductByQuery("Product.name = " + name, null); println("Product: " + product.getName() + ", price: " + product.getPrice());</pre>	<pre>Load Product Product: Networking, price: 320.0 *** Terminated, elapsed time: 391 ms ***</pre>

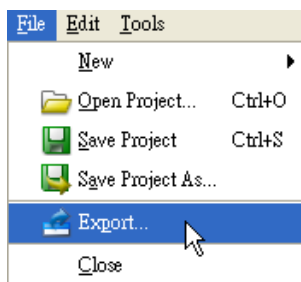
<p>LoadByQualifier Name</p> <pre>String name = "Steven Black"; Customer customer = CustomerFactory.loadByName(name); println("Name: " + customer.getName()); println("Contact Phone: " + customer.getPhone()); println("Email: " + customer.getEmail());</pre>	<p>LoadByQualifier Name</p> <pre>Name: Steven Black Contact Phone: 85412367 Email: steven.black@yahoo.com.us *** Terminated, elapsed time: 344 ms ***</pre>
<p>Update Product</p> <pre>PersistentSession session = CartPersistentManager.instance().getSession(); PersistentTransaction t = session.beginTransaction(); try { Product product = ProductFactory.loadProductByORMID(6); product.setUnitPrice(320); product.save(); t.commit(); } catch (Exception e) { t.rollback(); session.close(); }</pre>	<p>Update Product</p> <pre>*** Terminated, elapsed time: 296 ms ***</pre>
<p>Delete Product1</p> <pre>PersistentSession session = CartPersistentManager.instance().getSession(); PersistentTransaction t = session.beginTransaction(); try { Product product = ProductFactory.loadProductByORMID(10); product.delete(); t.commit(); } catch (Exception e) { t.rollback(); session.close(); }</pre>	<p>Delete Product</p> <pre>*** Terminated, elapsed time: 375 ms ***</pre>

Exporting SQL Statements and Code

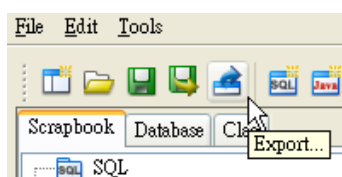
DB Visual ARCHITECT SQL (DB-VA SQL) allows you to export the SQL statements and code for further use and backup. In order to export file, you have to open the scrapbook that you want to export, and then you can export the scrapbook to a file with the extension corresponding to the type of scrapbook.

You can activate the export facility in one of the two ways:

- On the menu, click **File > Export...**



- On the toolbar, click the **Export...** icon.



After activated the Export facility, a **Save** dialog box is displayed, you can specify the location to save the exported file.

