SQLC

SQL – Java bridge generator

SQL Compiler for Java

Generates bridge classes from database and statement metadata

http://www.hammurapi.biz

Pavel Vlasov, January 2005

SQLC

SQL – Java bridge generator

Overview

- SQLC (**SQL C**ompiler) generates Java classes from SQL statements using statement and table metadata obtained from the database.
- It is implemented as an Ant task.
- The primary goal of SQLC is to decouple Java code from SQL (or any other QL).

SQLC

SQL – Java bridge generator

Benefits

• Classes are generated from metadata obtained from the target database. This approach helps to avoid situations when somebody invokes setString(1, "Hello") for numeric field.

• Generation at build time ensures that the database and the code are in sync. E.g. you have DEV and PROD database servers and application servers. You created a new table in DEV, but by some reason this change wasn't moved to PROD database. With SQLC this problem will be discovered during build time - the build will fail. Otherwise it would be discovered only in runtime. If the forgotten table participates in a rarely used use-case the problem may be discovered months after the code move when development team is already dismissed. If it happens, say, during year close and it will be A REALLY BIG PROBLEM.

• Probability of connection leakage in SQLC is much less comparing to traditional JDBC-based database programming. The only scenario of connection leakage in SQLC is obtaining database-backed collection, and not iterating over it till the end of connection. In this scenario iterator, which holds database resources, shall be explicitly closed or it will be closed in finalize().

• Decoupling of SQL from Java gives the following benefits:

  • It encourages usage of parameterized statements, which are believed to be more performant than queries assembled in runtime.

  • Java and SQL can be owned by different teams. SQL tuning can be performed without touching Java code.

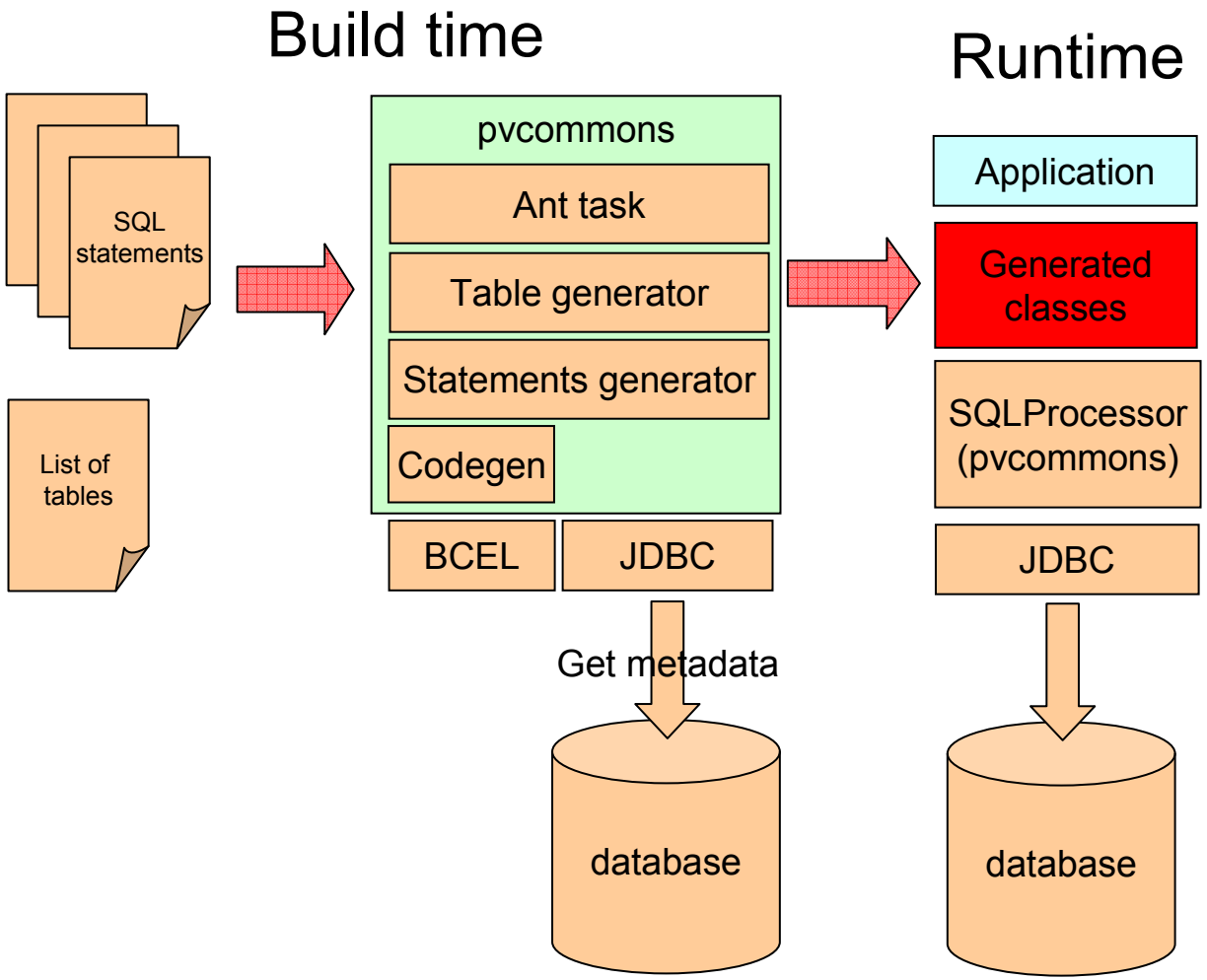
  • The same Java code may be used with different databases as long as compiled statements take the same number of parameters and return compatible results.

  • It makes easier to perform SQL code reviews - both manual and automated.

SQLC

SQL – Java bridge generator

Architecture



- Table generator – uses table metadata to create standard select, update and delete statements for table, indexes and foreign keys.
- Statements generator – generates engine methods for updates and engine methods plus interface/implementation to represent results for queries
- Generated classes – generated DAO (engine class) and DTO (interfaces and implementations).
- SQLProcessor – encapsulates JDBC resource management and projection patterns.

SQLC

SQL – Java bridge generator

Queries

```
<query
    name="ViolationJoined"
    description="Violations for result with all needed data"
>
    SELECT
        V.INSPECTOR,
        V.LINE,
        V.COL,
        V.SOURCE_ID,
        C.PATH || V.SIGNATURE_POSTFIX AS SIGNATURE,
        M.MESSAGE_VALUE AS MESSAGE,
        C.PATH AS SOURCE_URL
    FROM
        VIOLATION V
        LEFT JOIN MESSAGE M ON V.MESSAGE_ID=M.ID
        LEFT JOIN COMPILATION_UNIT C ON V.SOURCE_ID=C.ID
    WHERE
        V.RESULT_ID=? AND V.VIOLATION_TYPE=0
</query>
```

Interface

- ViolationJoined
  - getInspector()
  - getMessage()
  - getSignature()
  - setInspector(String)
  - setMessage(String)
  - setSignature(String)

Implementation

- ViolationJoinedImpl
  - project(SQLProcessor, String, Parameterizer)
  - project(SQLProcessor, String, Parameterizer, Collection)
  - ViolationJoinedImpl()
  - ViolationJoinedImpl(boolean)
  - ViolationJoinedImpl(ResultSet)
  - getCol()
  - getInspector()
  - getLine()
  - getMessage()
  - getSignature()
  - getSourceId()
  - getSourceUrl()
  - setCol(int)
  - setInspector(String)
  - setLine(int)
  - setMessage(String)
  - setSignature(String)
  - setSourceId(Integer)
  - setSourceUrl(String)

For a query SQLC generates

- Interface,
- Implementation class
- Engine methods

Engine methods

- getViolationJoined(int)
- getViolationJoined(int, Converter)
- getViolationJoined(int, Class)
- getViolationJoined(int, Collection)
- getViolationJoined(int, Collection, Converter)
- getViolationJoined(int, Collection, Class)
- processViolationJoined(int, RowProcessor)

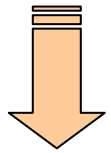
SQLC

SQL – Java bridge generator

Updates

SQLC generates one engine method per update

```
<update
    name="DeleteOldReports"
    description="Deletes reports with the same name other than this one. Parameters: id, name"
>
    <![CDATA[DELETE FROM REPORT WHERE ID<>? AND NAME=?]]>
</update>
```



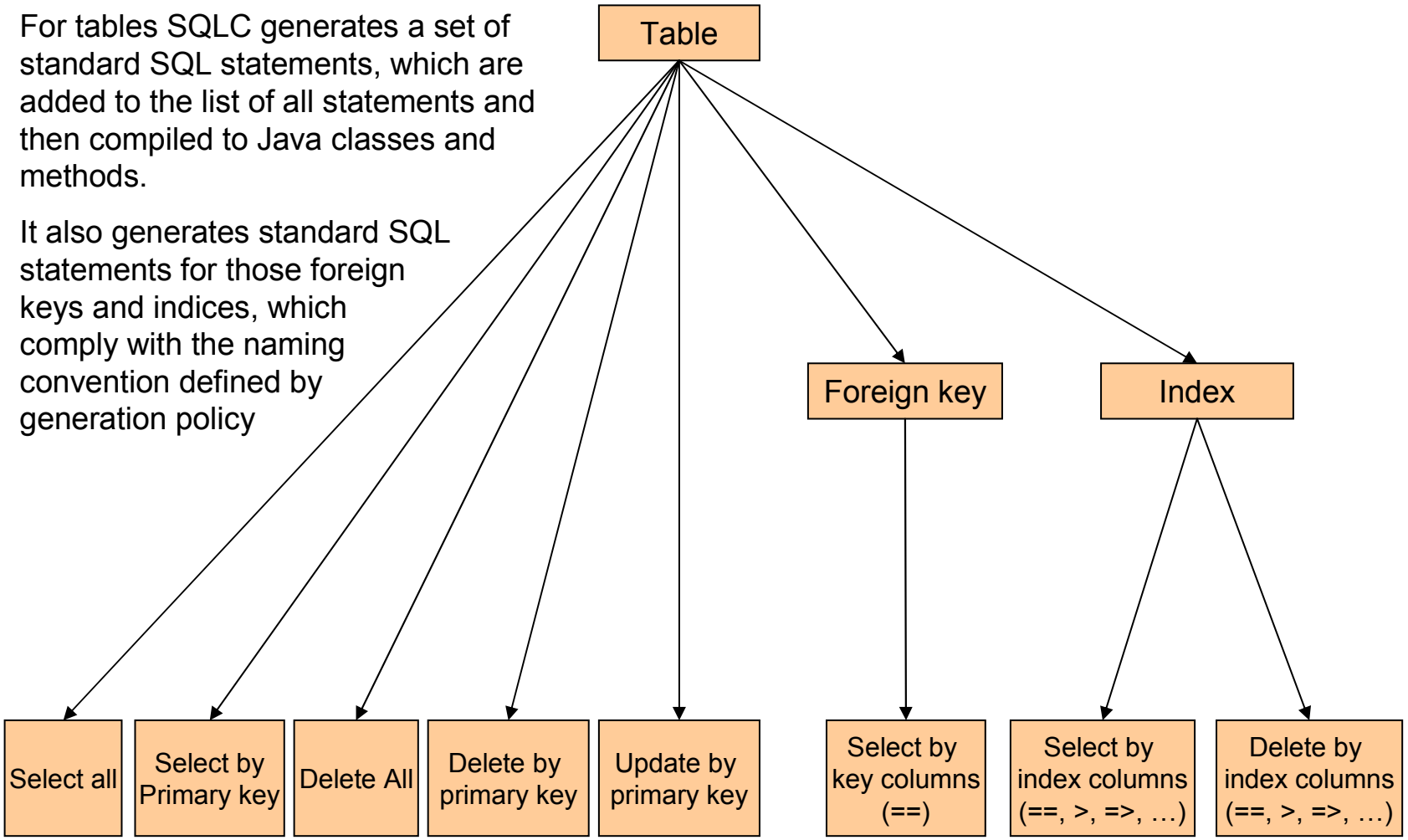
SQLC

SQL – Java bridge generator

Tables

For tables SQLC generates a set of standard SQL statements, which are added to the list of all statements and then compiled to Java classes and methods.

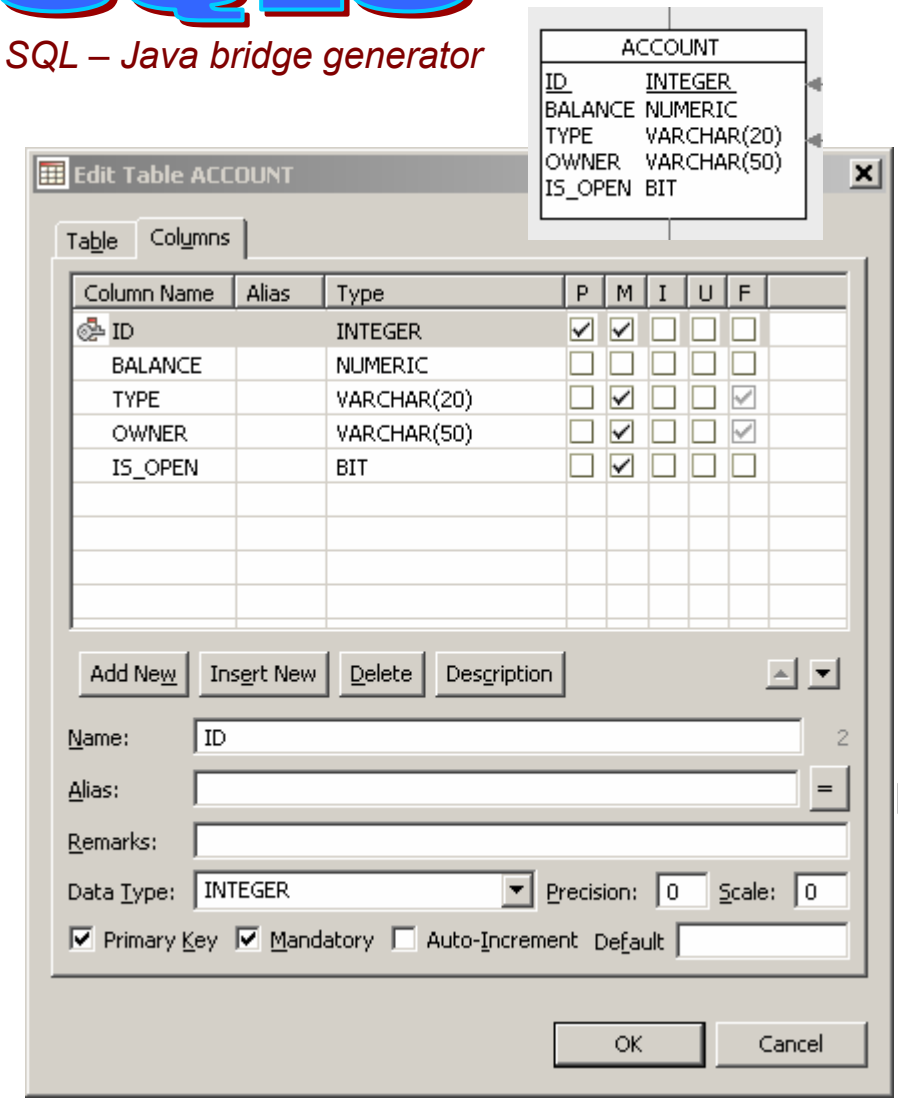
It also generates standard SQL statements for those foreign keys and indices, which comply with the naming convention defined by generation policy



SQLC

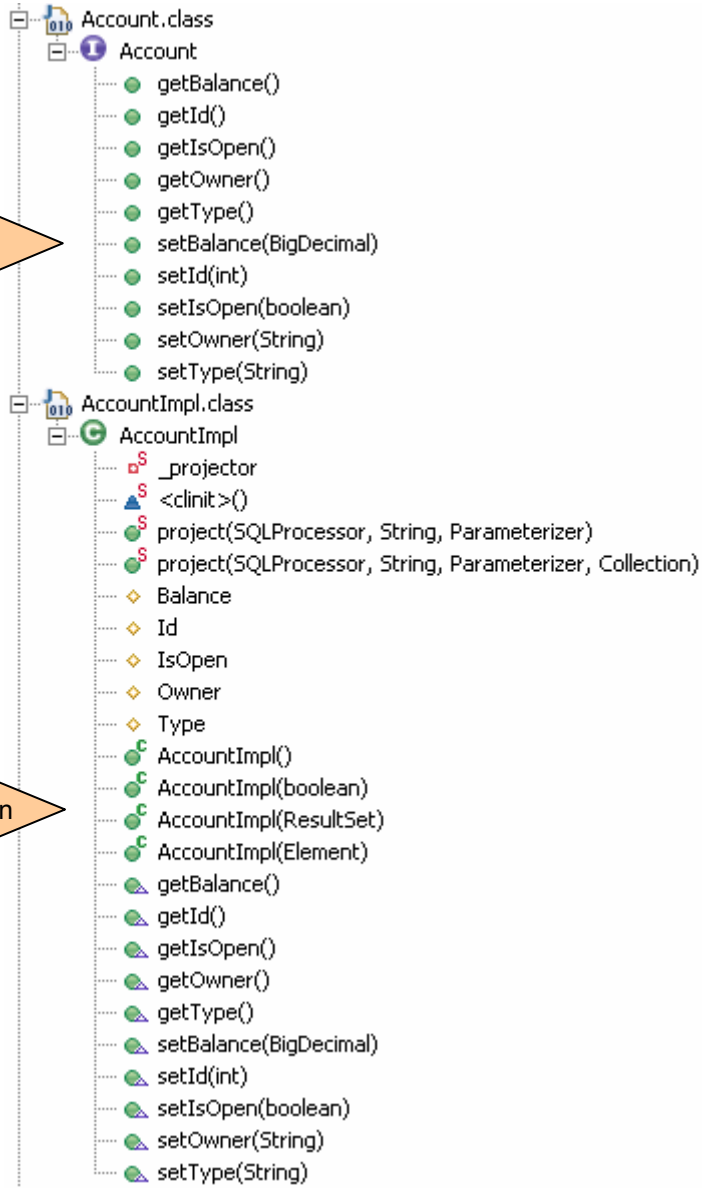
SQL – Java bridge generator

Table – interface



interface

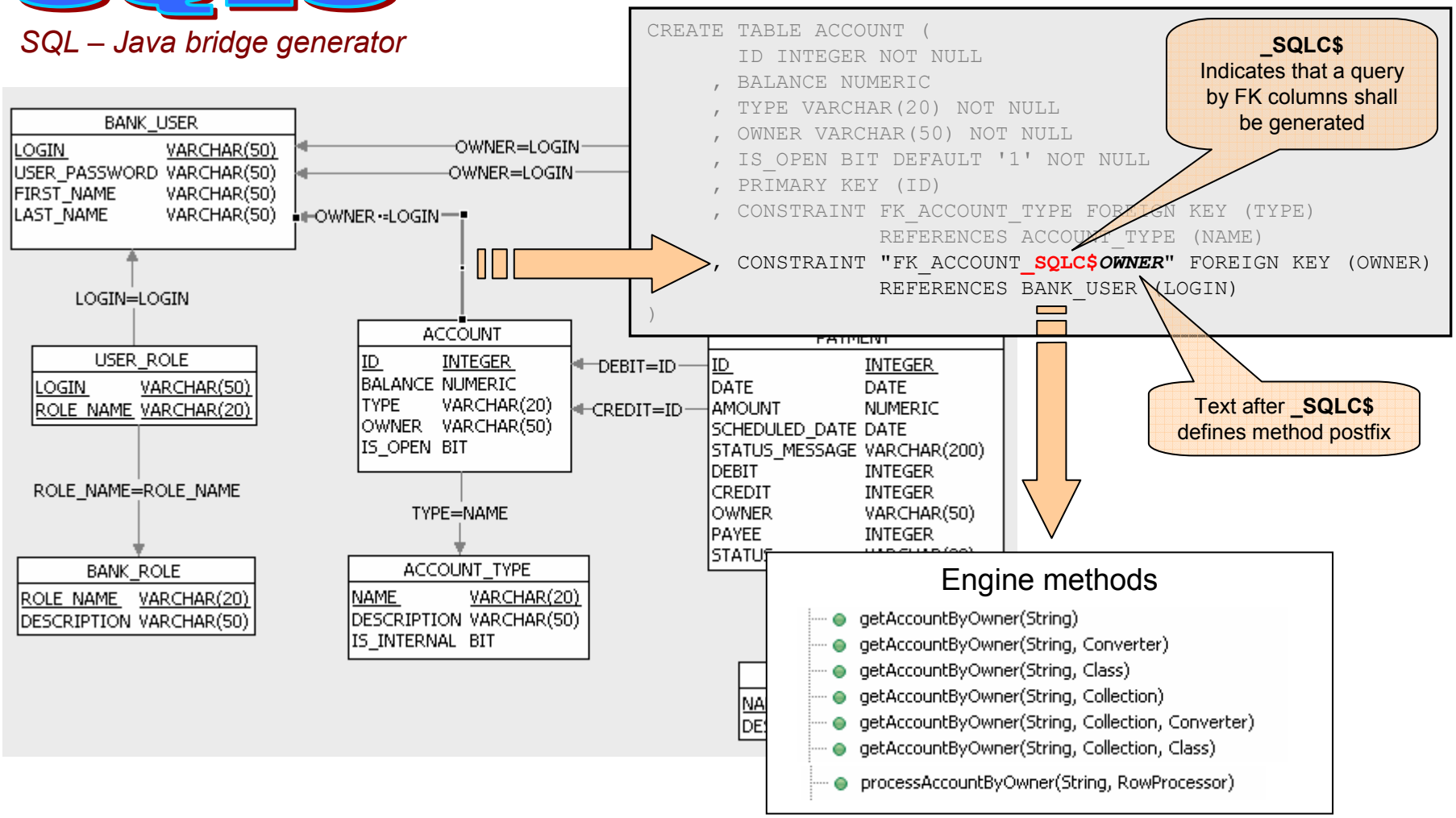
implementation



SQLC

SQL – Java bridge generator

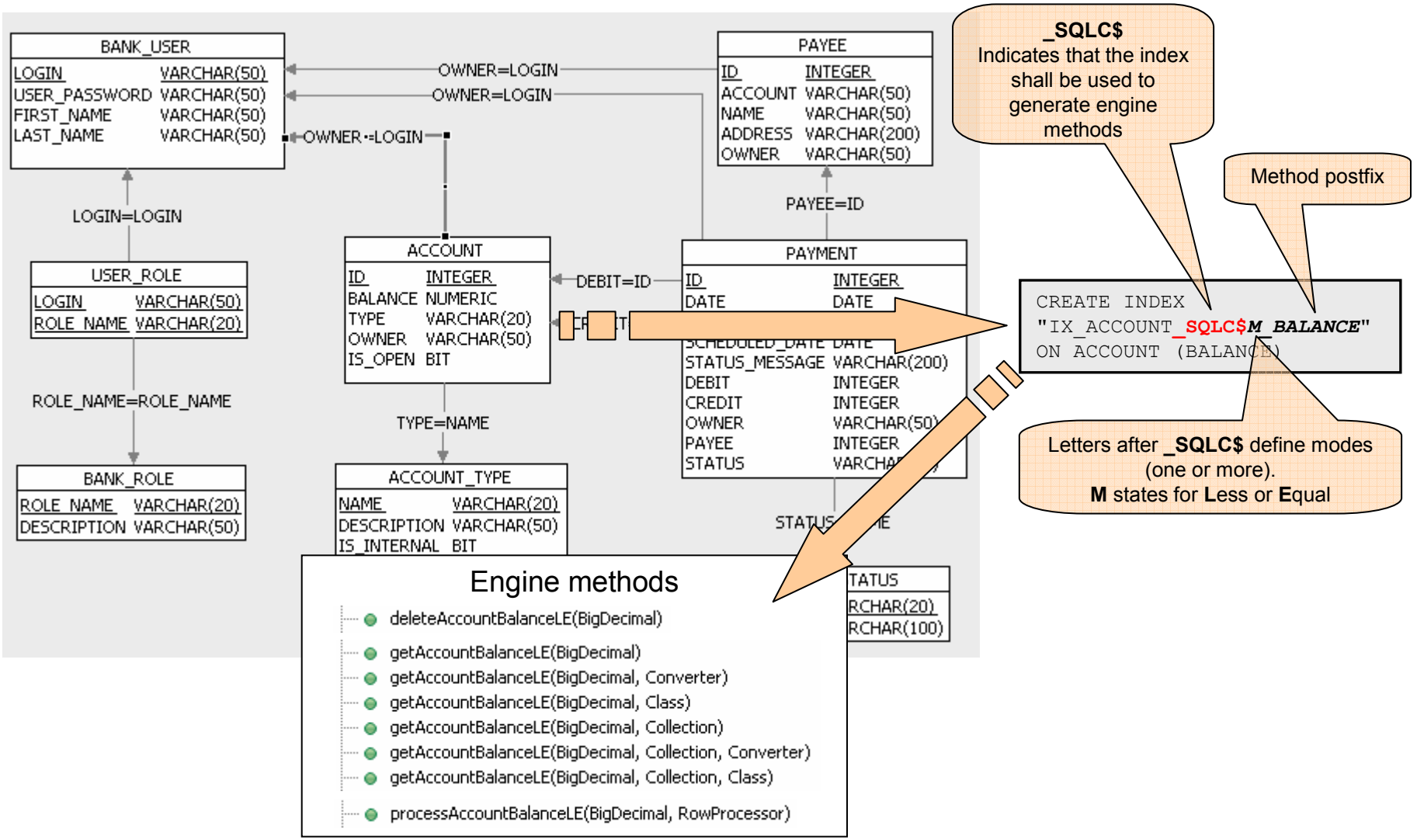
Table - Foreign keys



SQLC

SQL – Java bridge generator

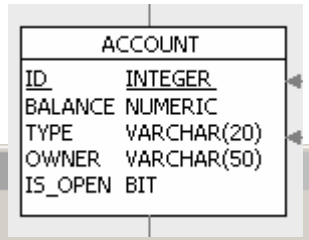
Table - Indices

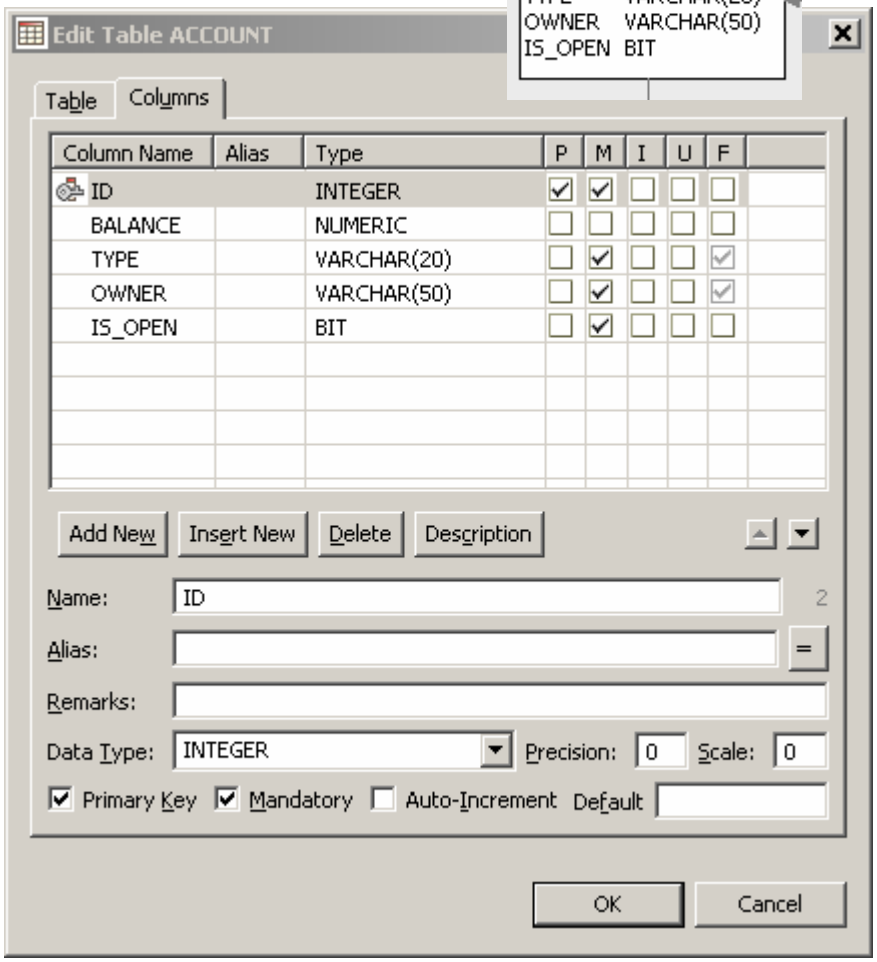


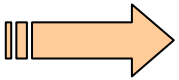
SQLC

SQL – Java bridge generator

Table – engine methods







Engine methods

- deleteAccount()
- deleteAccount(int)
- deleteAccountBalanceLE(BigDecimal)
- getAccount()
- getAccount(Converter)
- getAccount(int)
- getAccount(int, Class)
- getAccount(Class)
- getAccount(Collection)
- getAccount(Collection, Converter)
- getAccount(Collection, Class)
- getAccountBalanceLE(BigDecimal)
- getAccountBalanceLE(BigDecimal, Converter)
- getAccountBalanceLE(BigDecimal, Class)
- getAccountBalanceLE(BigDecimal, Collection)
- getAccountBalanceLE(BigDecimal, Collection, Converter)
- getAccountBalanceLE(BigDecimal, Collection, Class)
- getAccountByOwner(String)
- getAccountByOwner(String, Converter)
- getAccountByOwner(String, Class)
- getAccountByOwner(String, Collection)
- getAccountByOwner(String, Collection, Converter)
- getAccountByOwner(String, Collection, Class)
- insertAccount(Account)
- insertAccount(int, BigDecimal, String, String, boolean)
- processAccount(RowProcessor)
- processAccountBalanceLE(BigDecimal, RowProcessor)
- processAccountByOwner(String, RowProcessor)
- updateAccount(Account)

SQLC

SQL – Java bridge generator

Interface hierarchy

- SQLC generates inerfaces to represent resultsets.
- There can be multiple queries returning same set of columns. SQLC reuses already defined interfaces.
- You can explicitly suggest SQLC to reuse existing interfaces with nested ***interface*** element.
- SQLC creates interface hierarchy by discovering common methods in interfaces.
- SQLC discovers "Common denominator" interfaces.

SQLC

SQL – Java bridge generator

Implementations

- Dumb and Smart implementations
- Both implement equals(), hashCode(), toString() and toDom()
- Dumb implementations are generated for queries. `smart` attribute in `<query>` forces generation of smart implementation.
- Smart implementations are generated for tables.
- Smart implementations keep track of changes and construct insert/update/delete clauses dynamically.
- Smart implementations have fromDom(org.w3c.dom.Element) and constructor from org.w3c.dom.Element
- Engine methods delegate inserts/updates to Smart implementations.

SQLC

SQL – Java bridge generator

Advanced features

- Classes generated by SQLC can be subclassed to provide additional functionality.
- Projection can be done to SQLC-generated classes, their subclasses, or custom classes which has constructors taking SQLC-generated classes.
- If class being projected to implements *DataAccessObject* interface then it will be “infected” with SQLProcessor during projection, which allows it to keep link with the database to perform additional operations when needed.
- Conversion of database names to Java names in governed by implementations of com.pavelvlasov.sql.metadata.GenerationPolicy interface.
- There is a default GenerationPolicy implementation shipped with SQLC.
- This implementation can be subclasses or a new implementation can be written from scratch to customize code generation to fit organization’s database modeling standards.

SQLC

SQL – Java bridge generator

Nesting SQLC tasks

SQLC tasks can be nested, which allows to implement module-submodule relationship.

Nested tasks inherit

- Database connection,
- Output directory,
- Imported and generated interfaces

Main task

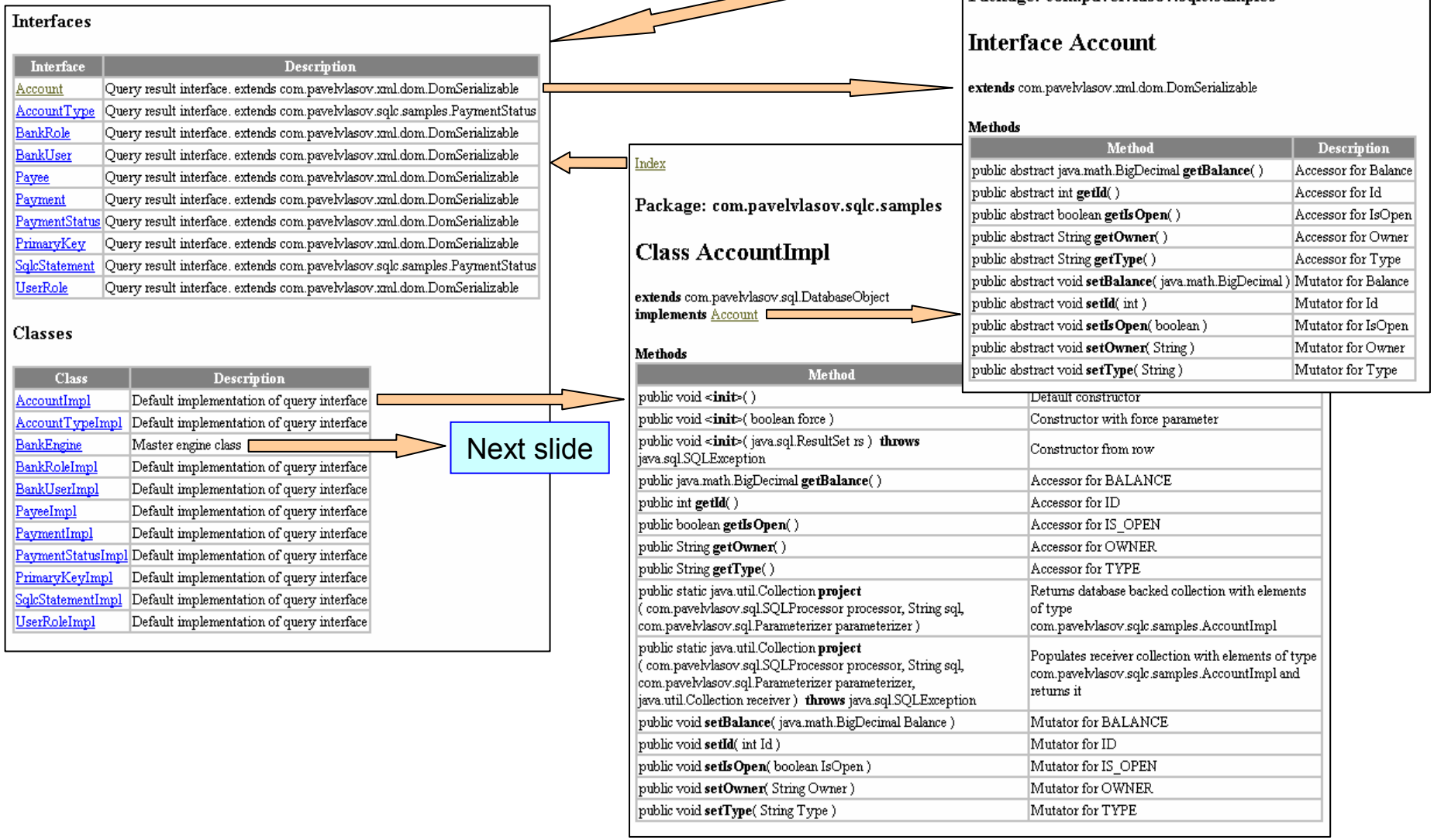
Nested task generates engine with methods specific for payment processing

```
<sqlc
    script="src/com/pavelvlasov/sqlc/samples/Bank.sql"
    dir="build/sqlc_generated"
    docDir="build/sqlcDocXml"
    xmlDoc="yes"
    package="com.pavelvlasov.sqlc.samples"
    masterEngine="BankEngine"
>
    <table/>
    <dbstatements statementsQuery="SELECT * FROM SQLC_STATEMENT"/>

    <query name="AccountNumberSubjectToServiceCharge"><![CDATA[
        SELECT
            ID
        FROM
            ACCOUNT
        WHERE
            BALANCE > 0
            AND BALANCE < ?
            AND TYPE=?
     ]]></query>

    <sqlc
        docDir="build/sqlcDocXml.payment"
        xmlDoc="yes"
        package="com.pavelvlasov.sqlc.samples.payment"
        masterEngine="PaymentEngine"
    >
        <update
            name="RestoreCredit"
            description="Parameters: payment id two times">
            UPDATE
                ACCOUNT
            SET
                BALANCE=BALANCE-(SELECT AMOUNT FROM PAYMENT WHERE ID=?)
            WHERE
                ID=(SELECT CREDIT FROM PAYMENT WHERE ID=?)
        </update>
    </sqlc>
</sqlc>
```

SQLC

SQL – Java bridge generator

Documentation



Interfaces

Interface	Description
Account	Query result interface. extends com.pavelvlasov.xml.dom.DomSerializable
AccountType	Query result interface. extends com.pavelvlasov.sqlc.samples.PaymentStatus
BankRole	Query result interface. extends com.pavelvlasov.xml.dom.DomSerializable
BankUser	Query result interface. extends com.pavelvlasov.xml.dom.DomSerializable
Payee	Query result interface. extends com.pavelvlasov.xml.dom.DomSerializable
Payment	Query result interface. extends com.pavelvlasov.xml.dom.DomSerializable
PaymentStatus	Query result interface. extends com.pavelvlasov.xml.dom.DomSerializable
PrimaryKey	Query result interface. extends com.pavelvlasov.xml.dom.DomSerializable
SqlcStatement	Query result interface. extends com.pavelvlasov.sqlc.samples.PaymentStatus
UserRole	Query result interface. extends com.pavelvlasov.xml.dom.DomSerializable

Classes

Class	Description
AccountImpl	Default implementation of query interface
AccountTypeImpl	Default implementation of query interface
BankEngine	Master engine class
BankRoleImpl	Default implementation of query interface
BankUserImpl	Default implementation of query interface
PayeeImpl	Default implementation of query interface
PaymentImpl	Default implementation of query interface
PaymentStatusImpl	Default implementation of query interface
PrimaryKeyImpl	Default implementation of query interface
SqlcStatementImpl	Default implementation of query interface
UserRoleImpl	Default implementation of query interface

Next slide

Index

Package: com.pavelvlasov.sqlc.samples

Interface Account

extends com.pavelvlasov.xml.dom.DomSerializable

Methods

Method	Description
public abstract java.math.BigDecimal **getBalance**( )	Accessor for Balance
public abstract int **getId**( )	Accessor for Id
public abstract boolean **getIsOpen**( )	Accessor for IsOpen
public abstract String **getOwner**( )	Accessor for Owner
public abstract String **getType**( )	Accessor for Type
public abstract void **setBalance**(java.math.BigDecimal)	Mutator for Balance
public abstract void **setId**(int)	Mutator for Id
public abstract void **setIsOpen**(boolean)	Mutator for IsOpen
public abstract void **setOwner**(String)	Mutator for Owner
public abstract void **setType**(String)	Mutator for Type

Index

Package: com.pavelvlasov.sqlc.samples

Class AccountImpl

extends com.pavelvlasov.sql.DatabaseObject
implements Account

Methods

Method	
public void <**init**>( )	Default constructor
public void <**init**>(boolean force)	Constructor with force parameter
public void <**init**>(java.sql.ResultSet rs) **throws** java.sql.SQLException	Constructor from row
public java.math.BigDecimal **getBalance**( )	Accessor for BALANCE
public int **getId**( )	Accessor for ID
public boolean **getIsOpen**( )	Accessor for IS_OPEN
public String **getOwner**( )	Accessor for OWNER
public String **getType**( )	Accessor for TYPE
public static java.util.Collection **project** (com.pavelvlasov.sql.SQLProcessor processor, String sql, com.pavelvlasov.sql.Parameterizer parameterizer)	Returns database backed collection with elements of type com.pavelvlasov.sqlc.samples.AccountImpl
public static java.util.Collection **project** (com.pavelvlasov.sql.SQLProcessor processor, String sql, com.pavelvlasov.sql.Parameterizer parameterizer, java.util.Collection receiver) **throws** java.sql.SQLException	Populates receiver collection with elements of type com.pavelvlasov.sqlc.samples.AccountImpl and returns it
public void **setBalance**(java.math.BigDecimal Balance)	Mutator for BALANCE
public void **setId**(int Id)	Mutator for ID
public void **setIsOpen**(boolean IsOpen)	Mutator for IS_OPEN
public void **setOwner**(String Owner)	Mutator for OWNER
public void **setType**(String Type)	Mutator for TYPE

Documentation (ctd)

SQLC

SQL – Java bridge generator

Engine class documentation provides detailed description for each method.

Index

Package: com.pavelvlasov.sqlc.samples

Class BankEngine

Methods

Method	Description
public void <**init**>(com.pavelvlasov.sql.SQLProcessor processor)	Constructor
public int **accountCredit**(java.math.BigDecimal p1, int p2) **throws** java.sql.SQLException	Executes update and returns number of affected rows. Increases balance
public int **accountDebit**(java.math.BigDecimal p1, int p2) **throws** java.sql.SQLException	Executes update and returns number of affected rows. Decreases balance
public int **deleteAccount**( ) **throws** java.sql.SQLException	Executes update and returns number of affected rows. Deletes all records from ACCOUNT
public int **deleteAccount**(int Id) **throws** java.sql.SQLException	Executes update and returns number of affected rows. Deletes by primary key from ACCOUNT
public int **deleteAccountBalanceLE**(java.math.BigDecimal Balance) **throws** java.sql.SQLException	Executes update and returns number of affected rows. Deletes row(s) with less or equal index value(s): BALANCE
public int **deleteAccountType**( ) **throws** java.sql.SQLException	Executes update and returns number of affected rows. Deletes all records from ACCOUNT_TYPE
public java.util.Collection **getAccount**( )	Executes query, returns database backed collection with elements of type com.pavelvlasov.sqlc.samples.Account. Selects all rows from ACCOUNT
public java.util.Collection **getAccount**(com.pavelvlasov.util.Converter converter)	Executes query, returns database backed collection with elements of type com.pavelvlasov.sqlc.samples.Account. If converter is not null then elements will be results of conversion.Selects all rows from ACCOUNT
public java.util.Collection **getAccount**(Class targetClass)	Executes query, returns database backed collection with elements of type com.pavelvlasov.sqlc.samples.Account. If targetClass is not null then elements will be of targetClass typeSelects all rows from ACCOUNT
public java.util.Collection **getAccount**(java.util.Collection receiver) **throws** java.sql.SQLException	Executes query, populates collection with elements of type com.pavelvlasov.sqlc.samples.Account and returns populated collection. Selects all rows from ACCOUNT
public java.util.Collection **getAccount**(java.util.Collection receiver, com.pavelvlasov.util.Converter converter) **throws** java.sql.SQLException	Executes query, populates collection with elements of type com.pavelvlasov.sqlc.samples.Account and returns populated collection. If converter is not null then elements will be results of conversion. Selects all rows from ACCOUNT
public java.util.Collection **getAccount**(java.util.Collection receiver, Class targetClass) **throws** java.sql.SQLException	Executes query, populates collection with elements of type com.pavelvlasov.sqlc.samples.Account and returns populated collection. If targetClass is not null then elements will be of targetClass type. Selects all rows from ACCOUNT
public Account **getAccount**(int Id) **throws** java.sql.SQLException	Executes query and returns single object. Selects by primary key from ACCOUNT
public Account **getAccount**(int Id, Class targetClass) **throws** java.sql.SQLException	Executes query and returns single object of target class type. Target class must be return type compatible. Selects by primary key from ACCOUNT
public int **insertAccount**(int Id, java.math.BigDecimal Balance, String Type, String Owner, boolean IsOpen) **throws** java.sql.SQLException	Executes update and returns number of affected rows. Inserts new record into ACCOUNT
public int **insertAccount**(Account rowInterface) **throws** java.sql.SQLException	Executes update and returns number of affected rows. Inserts new record into ACCOUNT

SQLC

SQL – Java bridge generator

MDA – building on SQLC output

SQLC generates documentation in XML, which can be used for further code generation – e.g. creation of Struts forms.

```
<classes>
 <class
   description="Master engine class"
   fcn="com.pavelvlasov.sqlc.samples.BankEngine"
   name="BankEngine"
   package="com.pavelvlasov.sqlc.samples"/>
 <class
   description="Query result interface.  extends com.pavelvlasov.xml.dom.DomSerializable"
   fcn="com.pavelvlasov.sqlc.samples.Payee"
   interface="yes"
   name="Payee"
   package="com.pavelvlasov.sqlc.samples"/>
 <class
   description="Default implementation of query interface"
   fcn="com.pavelvlasov.sqlc.samples.PayeeImpl"
   name="PayeeImpl"
   package="com.pavelvlasov.sqlc.samples"/>
 <class
   description="Query result interface.  extends com.pavelvlasov.
   fcn="com.pavelvlasov.sqlc.samples.Account"
   interface="yes"
   name="Account"
   package="com.pavelvlasov.sqlc.samples"/>
```

Fragment of index.xml

Fragment of BankEngine.xml

```
  <method
    description="Executes query, returns database backed collection with elements of
type com.pavelvlasov.sqlc.samples.Payee. If converter is not null then elements will
be results of conversion.Selects all rows from PAYEE"
    name="getPayee"
    signature="public java.util.Collection getPayee(com.pavelvlasov.util.Converter
converter)">
    <modifier>public</modifier>
    <return
      fcn="java.util.Collection"
      name="Collection"
      package="java.util"/>
    <parameter
      name="converter">
      <type
        fcn="com.pavelvlasov.util.Converter"
        name="Converter"
        package="com.pavelvlasov.util"/>
    </parameter>
    <attribute
      name="element-type">com.pavelvlasov.sqlc.samples.Payee</attribute>
  </method>
```

Attributes provide extra information

SQLC

SQL – Java bridge generator

Resources

- Download: pvcommons on http://www.hammurapi.biz/hammurapi-biz/ef/xmenu/downloads.html
- Sample application: http://www.hammurapi.biz/downloads/sqlc-samples.zip
- Article (HTML): http://www.hammurapi.biz/hammurapi-biz/ef/xmenu/products/common/sqlc/sqlc.html
- Another article (PDF): http://www.hammurapi.biz/doc/SQLC-FD.pdf