Table of Contents

If you're an Oracle user, you'll know that it's possible to update views with insteadOf triggers. An insteadOf trigger will do the CRUD operation(s), and it will be called with new/changed values like any other trigger. It's like a procedure that does CRUD programatically. It's a simple concept but not available in all databases.

With JVx' DBStorage, it's super easy to use insteadOf triggers in your business logic independent of the database.

We have a short example for you. Imagine you have an activity table with columns: name, cost, valid_from, valid_to, etc. The table also has a foreign key to a contract. One contract has one or more activities:

Our GUI shows a list of all available activities and it should be possible to change the contract. If the contract is available, this would be a straightforward implementation because JVx supports everything out of the box. However, we want to insert a new contract if it isn't available without additional popups.

We'll use an insteadOf trigger to solve this problem.

In the above screenshot, the “New contract” wasn't available in the contract table, but we did the insert during inserting a new activity. Here's the whole code:

TestInsteadOf.java

```
/*
 * Copyright 2016 SIB Visions GmbH
 *
 * Licensed under the Apache License, Version 2.0 (the "License"); you
may not
 * use this file except in compliance with the License. You may obtain
a copy of
 * the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT
 * WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See
the
 * License for the specific language governing permissions and
limitations under
 * the License.
 *
 *
 * History
 *
 * 11.08.2016 - [JR] - creation
```

```
 */
package com.sibvisions.forum;

import jvx.rad.application.SimpleTestLauncher;
import jvx.rad.genui.UIFactoryManager;
import jvx.rad.genui.container.UIPanel;
import jvx.rad.genui.layout.UIBorderLayout;
import jvx.rad.type.bean.IBean;
import jvx.rad.ui.celleditor.ILinkedCellEditor;

import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;

import com.sibvisions.apps.components.NavigationTable;
import com.sibvisions.apps.server.util.LifeCycleUtil;
import com.sibvisions.rad.persist.StorageDataBook;
import com.sibvisions.rad.persist.event.IStorageListener;
import com.sibvisions.rad.persist.event.StorageEvent;
import com.sibvisions.rad.persist.jdbc.DBAccess;
import com.sibvisions.rad.persist.jdbc.DBStorage;
import com.sibvisions.rad.ui.swing.impl.SwingFactory;
import com.sibvisions.util.type.CommonUtil;

public class TestInsteadOf extends UIPanel
{
    /** the database access. */
    private DBAccess dba;

    /** the contract storage. */
    private DBStorage dbsContract;
    /** the activity storage. */
    private DBStorage dbsActivity;

    /**
     * Initializes the unit test.
     *
     * @throws Exception if initialization fails
     */
    @BeforeClass
    public static void beforeClass()
    {
        UIFactoryManager.getFactoryInstance(SwingFactory.class);
    }

    /**
     * Initializes the test case.
     *
     * @throws Exception if initialization fails
     */
    @Before
```

```
    public void before() throws Exception
    {
        dba =
DBAccess.getDBAccess("jdbc:oracle:thin:@localhost:1521:XE",
                                    "username", "password");
        dba.open();
    }

    /**
     * Cleanup the test case.
     */
    @After
    public void after()
    {
        CommonUtil.close(dbsActivity, dbsContract, dba);
    }

    /**
     * Tests application start with a custom UI.
     *
     * @throws Exception if app start fails
     */
    @Test
    public void testInsteadOf() throws Exception
    {
        DBStorage dbsActivities = new DBStorage();
        dbsActivities.setAutoLinkReference(false);
        dbsActivities.setFromClause("V_ACTIVITIES");
        dbsActivities.setWritebackTable("ACTIVITY");
        //manual autolink
        dbsActivities.createAutomaticLinkReference(new String[]
{"CONT_ID", "CONT_TITLE"},
                                                   "CONTRACT",
                                                   new String[] {"ID",
"TITLE"});
        dbsActivities.eventInsteadOfInsert().addListener(new
IStorageListener()
        {
            public void storageChanged(StorageEvent pStorageEvent)
throws Throwable
            {
                boolean bAutoCommit = dba.isAutoCommit();

                try
                {
                    dba.setAutoCommit(false);

                    DBStorage contract = getContract();

                    IBean newRecord = pStorageEvent.getNew();
```

```
                    //check if a new contract should be created for the
activity

                    if (newRecord.get("CONT_ID") == null)
                    {
                        IBean bnContract = contract.createEmptyBean();

                        bnContract.put("TITLE",
newRecord.get("CONT_TITLE"));

                        bnContract = contract.insert(bnContract);

                        //we need the new id in our activity record!
                        newRecord.put("CONT_ID", bnContract.get("ID"));
                    }

                    //updates newRecord, because we need the updated
columns (e.g. ID)
                    newRecord.putAll(getActivity().insert(newRecord));

                    dba.commit();
                }
                catch (Throwable th)
                {
                    dba.rollback();

                    throw th;
                }
                finally
                {
                    dba.setAutoCommit(bAutoCommit);
                }
            }
        });
        dbsActivities.setDBAccess(dba);
        dbsActivities.open();

        StorageDataBook sdbActivities = new
StorageDataBook(dbsActivities);
        sdbActivities.open();

((ILinkedCellEditor)sdbActivities.getRowDefinition().getColumnDefinitio
n("CONT_TITLE").
getDataType().getCellEditor()).setValidationEnabled(false);

        setLayout(new UIBorderLayout());

        NavigationTable nav = new NavigationTable();
        nav.setDataBook(sdbActivities);
        //show all columns
        //nav.setColumnView(new
```

```
ColumnView(sdbActivities.getRowDefinition().getColumnNames()));

        add(nav);

        new SimpleTestLauncher(this);
    }

    /**
     * Gets the contract storage.
     *
     * @return the contract storage.
     * @throws Exception if creation fails
     */
    public DBStorage getContract() throws Exception
    {
        if (dbsContract == null)
        {
            dbsContract = LifeCycleUtil.createWriteBackStorage(dba,
"CONTRACT");
        }

        return dbsContract;
    }

    /**
     * Gets the activity storage.
     *
     * @return the contract storage.
     * @throws Exception if creation fails
     */
    public DBStorage getActivity() throws Exception
    {
        if (dbsActivity == null)
        {
            dbsActivity = LifeCycleUtil.createWriteBackStorage(dba,
"ACTIVITY");
        }

        return dbsActivity;
    }

}   // TestInsteadOf
```

The important things are:

```
...setValidationEnabled(false);

eventInsteadOfInsert()
```

The first method allows inserting new contracts because the cell editor won't check for existing

contracts. The second method implements the insteadOf trigger.

The trigger itself checks if the contract ID is empty and creates a new contract in this case. The activity will be inserted with the new contract id. The eventInsteadOfUpdate is missing in our example, but it works the same way.

You'll find the implementation of SimpleTestLauncher here.

The database objects (Oracle):

create.sql

```
CREATE TABLE CONTRACT
(
  id    NUMBER(16) NOT NULL,
  title VARCHAR2(1000),
  state NUMBER(1) DEFAULT 1
)
;
COMMENT ON COLUMN CONTRACT.id
  IS 'PK';
COMMENT ON COLUMN CONTRACT.title
  IS 'Contract title';
COMMENT ON COLUMN CONTRACT.state
  IS 'Contract state';
ALTER TABLE CONTRACT
  ADD CONSTRAINT CONT_PK PRIMARY KEY (ID);

CREATE TABLE ACTIVITY
(
  id                  NUMBER(16) NOT NULL,
  cont_id             NUMBER(16),
  name                VARCHAR2(1000),
  cost                NUMBER(10,2),
  valid_from          DATE,
  valid_to            DATE,
  responsible_user_id NUMBER(16),
  validated           CHAR(1) DEFAULT 'Y'
)
;
COMMENT ON COLUMN ACTIVITY.id
  IS 'PK';
COMMENT ON COLUMN ACTIVITY.cont_id
  IS 'Contract FK';
COMMENT ON COLUMN ACTIVITY.name
  IS 'Activity name';
COMMENT ON COLUMN ACTIVITY.cost
  IS 'Activity costs';
COMMENT ON COLUMN ACTIVITY.valid_from
  IS 'Valid from';
COMMENT ON COLUMN ACTIVITY.valid_to
```

```
  IS 'Valid to';
COMMENT ON COLUMN ACTIVITY.responsible_user_id
  IS 'Responsible User FK';
COMMENT ON COLUMN ACTIVITY.validated
  IS 'Whether the activity is valid';
ALTER TABLE ACTIVITY
  ADD CONSTRAINT ACTI_PK PRIMARY KEY (ID);
ALTER TABLE ACTIVITY
  ADD CONSTRAINT ACTI_CONT_FK FOREIGN KEY (CONT_ID)
  REFERENCES CONTRACT (ID);
ALTER TABLE ACTIVITY
  ADD CONSTRAINT ACTI_RESP_USER_FK FOREIGN KEY (RESPONSIBLE_USER_ID)
  REFERENCES USERS (ID);

CREATE SEQUENCE SEQ_ACTIVITY_ID
minvalue 1
maxvalue 9999999999999999999999999999
START WITH 41
INCREMENT BY 1
cache 20
ORDER;

CREATE SEQUENCE SEQ_CONTRACT_ID
minvalue 1
maxvalue 9999999999999999999999999999
START WITH 21
INCREMENT BY 1
cache 20
ORDER;

CREATE OR REPLACE force VIEW v_activities AS
SELECT a.id
      ,a.cont_id
      ,c.title cont_title
      ,a.name
      ,a.cost
      ,a.valid_from
      ,a.valid_to
  FROM activity a
      ,contract c
 WHERE a.cont_id = c.id;

CREATE OR REPLACE TRIGGER TR_ACTIVITY_BR_IU
  BEFORE INSERT OR UPDATE ON activity
  FOR each ROW
BEGIN

  IF (:NEW.id IS NULL) THEN
    SELECT seq_activity_id.NEXTVAL INTO :NEW.id FROM dual;
  END IF;
```

```
  IF (:NEW.validated = 'N') THEN

    IF (:NEW.valid_from IS NULL) THEN
      raise_application_error(-20000, 'Valid from can''t be null!');
    END IF;

  END IF;

END TR_ACTIVITY_BR_IU;
/

CREATE OR REPLACE TRIGGER TR_CONTRACT_BR_IU
  BEFORE INSERT OR UPDATE ON contract
  FOR each ROW
BEGIN

  IF (:NEW.id IS NULL) THEN
    SELECT seq_contract_id.NEXTVAL INTO :NEW.id FROM dual;
  END IF;

  IF (:NEW.state = 4) THEN

    UPDATE activity
       SET validated = 'N'
     WHERE cont_id = id
       AND validated = 'Y';

  END IF;

END TR_CONTRACT_BR_IU;
/

prompt Disabling triggers FOR CONTRACT...
ALTER TABLE CONTRACT disable ALL triggers;
prompt Disabling triggers FOR ACTIVITY...
ALTER TABLE ACTIVITY disable ALL triggers;
prompt Disabling FOREIGN KEY constraints FOR ACTIVITY...
ALTER TABLE ACTIVITY disable CONSTRAINT ACTI_CONT_FK;
ALTER TABLE ACTIVITY disable CONSTRAINT ACTI_RESP_USER_FK;
prompt Loading CONTRACT...
INSERT INTO CONTRACT (id, title, state)
VALUES (1, 'Air', 1);
COMMIT;
prompt 1 records loaded
prompt Loading ACTIVITY...
INSERT INTO ACTIVITY (id, cont_id, name, cost, valid_from, valid_to,
responsible_user_id, validated)
VALUES (1, 1, 'Part 1', .2, NULL, NULL, NULL, 'Y');
COMMIT;
prompt 1 records loaded
prompt Enabling FOREIGN KEY constraints FOR ACTIVITY...
```

```
ALTER TABLE ACTIVITY enable CONSTRAINT ACTI_CONT_FK;
ALTER TABLE ACTIVITY enable CONSTRAINT ACTI_RESP_USER_FK;
prompt Enabling triggers FOR CONTRACT...
ALTER TABLE CONTRACT enable ALL triggers;
prompt Enabling triggers FOR ACTIVITY...
ALTER TABLE ACTIVITY enable ALL triggers;
```