OPITZ CONSULTING

Überraschend mehr Möglichkeiten

Orcas: Continuous Delivery for Databases

A Framework for Continuous Delivery in Database Environments

Whitepaper

Orcas: Continuous Delivery for Databases

A Framework for Continuous Delivery in Database Environments

Author

Dr. Olaf Jessensky

Contact

Torsten Jaeschke
OPITZ CONSULTING Deutschland GmbH
Senior Consultant
torsten.jaeschke@opitz-consulting.com

Imprint

OPITZ CONSULTING Deutschland GmbH
Kirchstr. 6
D-51647 Gummersbach (Germany)
+49 (0)2261 6001-0
info@opitz-consulting.com

Table of contents

Preamble

One of the main concerns in modern software development is building and testing applications as easily as possible and deploying them to different target environments. To achieve this, there are procedures summarized among experts by the term Continuous Integration (CI) [1].

For projects in Java/JEE environments the result is usually a jar, war or ear file installed on an appropriately configurable target system, typically on a JEE application server. The trend is to deliver containerized applications runnable without an external application server. In both cases, the build process is set up to check out the source files of the application from a source code repository such as SVN or Git, and build, test and possibly deploy them using a build tool such as Maven, Ant or Gradle.

This build process is typically controlled by a CI server such as Hudson or Jenkins. The CI server can be configured in a way that a new build is initiated after each commit of a change in the repository, and the result is immediately available in a test environment.

In this whitepaper, I introduce you to the framework Orcas, which can significantly simplify the delivery of software solutions on the database level.

When implementing continuous delivery in a build pipeline, databases can be problem, in particular, classic fixed-schema relational databases. As the application evolves, the database schema must adapt to it to match the appropriate version of the application.

Usually, changing a schema also requires migrating the corresponding data. Test and production data are an important asset, and integrity must be ensured in the schema evolution as well. It must also be considered that an application may have a different state on different environments. All existing actual states must therefore be transferred to a new target state.

Continuous delivery often maintains more environments than traditional approaches. Therefore, the need for automatic migration is increasing. At this point our framework Orcas [2] comes on the scene.

1 What does Orcas do?

Orcas makes it possible to describe the state of tables in an Oracle database with a SQL-like syntax. The framework finds the delta of a given target schema in the database to the target state defined in the source code. As result, the framework transfers the current state to the target state ensuring that all data is retained. This closes the gap that has previously made continuous integration difficult for the database. In addition, developers using Orcas can efficiently manage the source code for the database in their source code repository [3].

This approach does not cover problems with migrating data to new tables or table columns. Here, Orcas offers the possibility of executing SQL scripts as so-called one-time scripts: The program remembers which scripts were executed in a certain schema and executes only newly added scripts.

At this point, the difference between Orcas and commercially available tools such as Liquibase [4] or Flyway [5] becomes clear. While these generally only manage scripts for single use, to migrate from a specific actual state to a specific target state, only the target state is described in Orcas, whereas the delta to the actual state is determined automatically by Orcas itself.

The advantages are

- greater flexibility due to the possibility of performing a deployment independent of the actual state of the target system
- and the ability to see the database development status that in Liquibase and Flyway derives only from the sum of all migration scripts.

The price for this advantages is the limitation of Orcas to the Oracle RDBMS.

2 A description language for database objects

The approach followed with Orcas involves defining a domain-specific language that describes the database objects to be realized. The syntax of this language is as close as possible to SQL, with some language variations and limitations.

The following example shows a simple table script for the "`orders`" table:

```
create table orders
(
  ordr_id         number(15)                not null,
  version         number(15)   default "0"  not null,
  bpar_id         number(15)                not null,
  orderdate         date                    not null,
  tracking_number   varchar2(20)            not null,

  constraint ordr_pk primary key (ordr_id),
  constraint ordr_uc unique (tracking_number),
  constraint ordr_bpar_fk foreign key (bpar_id)
  references business_partners (bpar_id)
);
```

All data fields and objects related to the "`orders`" table are listed within the `create table` command. The example demonstrates the simple SQL-like syntax for a primary key, a unique constraint, and a foreign key.

3 Flexible extensibility of the syntax

One of the outstanding features of Orcas is the ability to define custom syntax extensions. In this way, the framework can be adapted to the requirements of the respective project with little effort. As a simple example, I would like to introduce the domain extension contained in Orcas, which can be configured individually and which is extremely useful for the uniform handling of tables in a project.

A table of the domain `id_table` should always automatically receive a numeric PK column with the naming convention `<table_name> _id`. The associated PK constraint should have the name `<table_name>_pk`. For this, it is only specified in the table script that the table belongs to the Table Domain `id_table`.

Here is an example:

```
create table tab_a domain id_table
(
  somevalue         varchar2(100)
);
```

The next code excerpt shows how easy this functionality can be implemented.

The first part of the example shows the definition of a column domain that acts at the level of table columns. This is used in the table domain below.

```
define column domain pk_column generate-primary-key
(constraint-name(table-name || "_pk"))
(
  number(10) not null
);
define table domain id_table
(
  add column column-name
(table-name||"_"||column-name)(id domain pk_column)
);
```

The list of use cases for such enhancements ranges from the PK example shown above, through the creation of monitoring fields for the user who makes a change and the time of a change, up to the implementation of historization solutions..

4 Comparison with the database

For the comparison with the actual state of the DB schema to be processed, Orcas has an implemented Java logic that determines the current state from the Oracle database's data dictionary and compares it with the target state. Through this SQL commands are generated that establish the target state and which can be executed at the end of the synchronization on the database.

After the synchronization and the script generation a fully automatic deployment of the changes to the target database is scheduled by default. This procedure can be easily integrated into a continuous delivery pipeline.

Optionally, Orcas can be configured to not execute the resulting SQL script directly but to store it in the file system. In that case, the user can check the script before execution. Often, this approach is chosen when changes to a productive system are only allowed to specifically authorized persons and if they need to be accurately documented.

The resulting script describes the path from a certain actual state to the desired state and thus corresponds in the result to what you also get with Liquibase or Flyway. The difference is that the developer with Orcas maintains the database objects like normal source code and can automatically generate the delta script for a specific deployment.

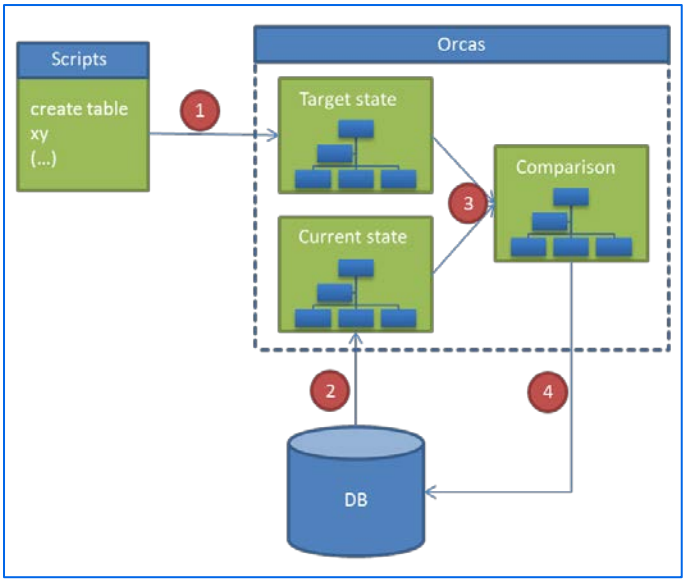

Illustration 1: Database object state comparison

5 Build process integration

The incorporation of orcas into a continuous build process is done using one of the supported build tools. Currently, these are Maven, Gradle and Ant. Thus, Orcas can be integrated into an existing CI build, which is implemented for example in Maven. It makes sense for the user to build the build process by running database tables first and then other database objects, such as views and packages, that reference tables processed before. Most of these objects have no application status like the tables, but can be recompiled like normal source code in the database. This category of objects is deployed by Orcas by executing the stored SQL scripts.

Afterwards, one-time scripts for data migrations or similar can be executed as required. For projects that have used Liquibase so far, Orcas has created the opportunity to call Liquibase as part of an Orcas build. Thus, Liquibase is still available to the developer for the management of one-time scripts. At the same time, the abilities of Orcas can be used to adjust the schema state.

6 Getting started with Orcas

For its functionality, Orcas requires Java (from version 8), at least one to the database environment suitable JDBC driver and depending on the choice of the build tool, Maven (from version 3), Gradle (at least version 3.3) or the build tools Ant and Gradle (in the respective current version). For Maven and Gradle Orcas is also available as an artifact in the Maven Central Repository [6]. The above prerequisites must be met on every developer workstation as well as on every build server on which Orcas is meant to be used. For testing purposes, we provide a Vagrant configuration [7] that allows you to configure and start a VM. This contains a database installation in which Orcas is directly executable.

If you start a project from scratch, you usually start with the creation of Orcas scripts after defining and implementing or configuring the needed extensions. More often, developers are in the situation of having to build on an existing DB schema. Since the syntax of Orcas differs slightly from SQL, this requires a reverse engineering. To do this, there is a task in Orcas for each supported build tool that translates the current state of a DB schema into Orcas scripts. Based upon these scripts one can continue the further development of the schema with Orcas.

7 Conclusion

Orcas enables developers to manage and evolve the definition of Oracle databases in the development process just like common source code. In this way, it provides the opportunity to apply the concepts of continuous delivery and continuous integration to the database sector. At the same time, the included syntax extensions provide a powerful tool that facilitates the management of data models. With this, the framework goes far beyond the possibilities of Flyway and Liquibase.

Orcas was released under Github [8] as open source. Everyone is invited to use it and participate in its development.

8 References

[1] http://www.thoughtworks.com/continuous-integration

[2] http://opitzconsulting.github.io/orcas/

[3] http://databaserefactoring.com/

[4] http://www.liquibase.org/

[5] http://flywaydb.org/

[6] http://www.mvnrepository.com/artifact/com.opitzconsulting.orcas

[7] https://www.vagrantup.com/

[8] https://github.com/opitzconsulting/orcas

9 Continuous Delivery by OPITZ CONSULTING

It often takes time from confirming activation of a change in the source code until the release of a software system in production. Changes are tested, configurations changed, patches installed and databank schemes updated. The results are long release cycles and anguish before the next release. At the same time the expectations of the professional users increase.

Short Change Cycles

For IT (Information Technology) to be a partner in a company, the software systems must change as fast as the business itself. Continuous Delivery enables short, agile iteration by reducing the timespan from the demand of productive support using your software from weeks/months to days/hours.

Automation via build pipeline

Thanks to modern development and test concepts as well as a high degree of automation, you will gain more safety and quality in your supply chain. A build pipeline takes care of the automated workflow by visualising the path every change must take; like a production line in a factory. The build pipeline creates an overview of present and past code changes and their production stage. Once all stages of the build pipeline are completed, they can be uploaded by a mouse click in the production.

What benefits does Continuous Delivery provide?

- Fast and low-risk delivery of changes generate higher added value of IT
- Faster 'Time-to-Market' by using consistent processes without complex transfers from the development to the productive rollout in customer projects
- Better release quality by using an automatic and reproducible process and limiting the source of errors

Based on your individual roadmap, we support you from requirement analysis, introduction of tools, adjustment of build processes and procedure models (scrum Kanban) to developing an integrated Continuous Delivery platform.

Keynote Speech/Requirement Analysis

Learn about Continuous Delivery and what it means for you and together with our experts, define your implementation strategy. Become familiar with tools such as Vagrant, Docker and Ansible (from an architectural perspective) and see how these tools will become an essential component of your Build Pipeline.

Roadmap

We assist you find and understand the 'bottleneck' areas in your development and provisioning process. Building on this value stream analysis, we will suggest practices and tools which you can use to change your processes. Together we look which steps you can take on your own and which you can delegate to us.

Implementation

- Development of a build pipeline
- Integration of log management and application performance management into the build process
- Provision of environments based on application containers (docker) and virtual machine
- Managing data bases within continuous delivery

Further Information and Contact

Further (interesting) information on DevOps and Continuous Delivery is available on www.opitz-consulting.com/devops. Visit our homepage www.opitz-consulting.com to find additional information on our offers and services.