Continuous Delivery mit Orcas

Deployment von Oracle-Datenbanken in agilen Projekten

Dr. Olaf Jessensky
Senior Consultant

OPITZ CONSULTING Deutschland GmbH

APEX Connect 2016, Berlin, 26.04.2016

OPITZ CONSULTING

Menschen. Innovationen. Lösungen.

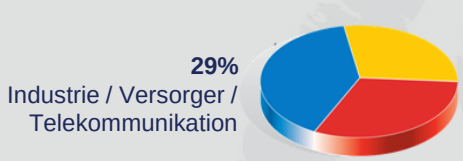
Mission

Wir entwickeln gemeinsam mit allen Branchen Lösungen, die dazu führen, dass sich diese Organisationen besser entwickeln als ihr Wettbewerb.

Unsere Dienstleistung erfolgt partnerschaftlich und ist auf eine langjährige Zusammenarbeit angelegt.

Märkte

- Branchenübergreifend
- Über 600 Kunden



29% Handel / Logistik / Dienstleistungen

29% Industrie / Versorger / Telekommunikation

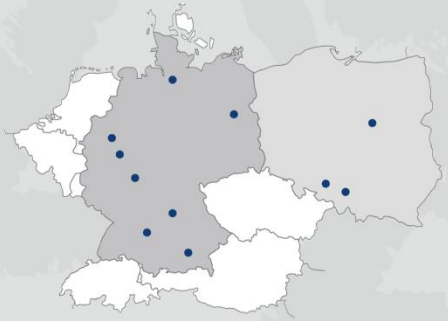
42% Öffentliche Auftraggeber / Banken und Versicherungen / Vereine und Verbände

Leistungsangebot

- Application Lifecycle Management
- IT-Beratung
- Business-Lösungen
- Managed Services
- Training und Coaching
- IT-Trends

Eckdaten

- Gründung 1990
- 400 Mitarbeiter
- 11 Standorte



Agenda

1. Agiles Datenbankdeployment
2. Funktionsweise von Orcas
3. Projektspezifische Anpassung
4. Arbeiten mit Orcas
5. Einsatzszenarien

1

Agiles Datenbankdeployment

Continuous Delivery

- **Einsatz in agilen Softwareprojekten**
- **Optimierung des Auslieferungsprozesses**
- **Continuous Delivery Pipeline**
- **Continuous Integration (CI)**
  - Bauen der Anwendung mit jeder Codeänderung
  - Permanente Lieferfähigkeit
  - Zentrales Repository, CI-Server
- **Testautomatisierung**
- **Continuous Deployment**
- **Integration von Datenbankskripten notwendig**

Deploymentstrategien für die Datenbank

- **Klassisches Vorgehen („old school“)**
  - Abnahme-DB als Referenz für die Produktion
  - Compare & Sync –Tools (Redgate, DBMaestro)
  - Keine Verwaltung der DB-Skripte im Sourcecode-Repository
  - DBAs sind die Besitzer der Datenbank
- **Agiles Vorgehen („new school“)**
  - Datenbank ist Teil der Anwendung
  - DB-Skripte sollen mit Anwendungssourcen verwaltet werden
  - Anwender sind Owner der Datenbank
  - Unterstützung durch Open Source Projekte (Liquibase, Flyway, Orcas)
  - Varianten: Delta-Skripte vs. Soll-Zustand

Eigenschaften von Orcas

- Beschreibung des Sollzustands der Datenbank
- Orcas ermittelt den Zustand des Zielschemas und erzeugt die passenden Delta-Skripte
- Sollzustand wird unabhängig vom Ist-Zustand erreicht
- Zu jeder Version ist der Zustand der DB direkt erkennbar
- Liquibase/Flyway: Datenbank ist die Summe aller Delta-Skripte
- Skripte verhalten sich wie Anwendungssourcen

2 Orcas Funktionsweise

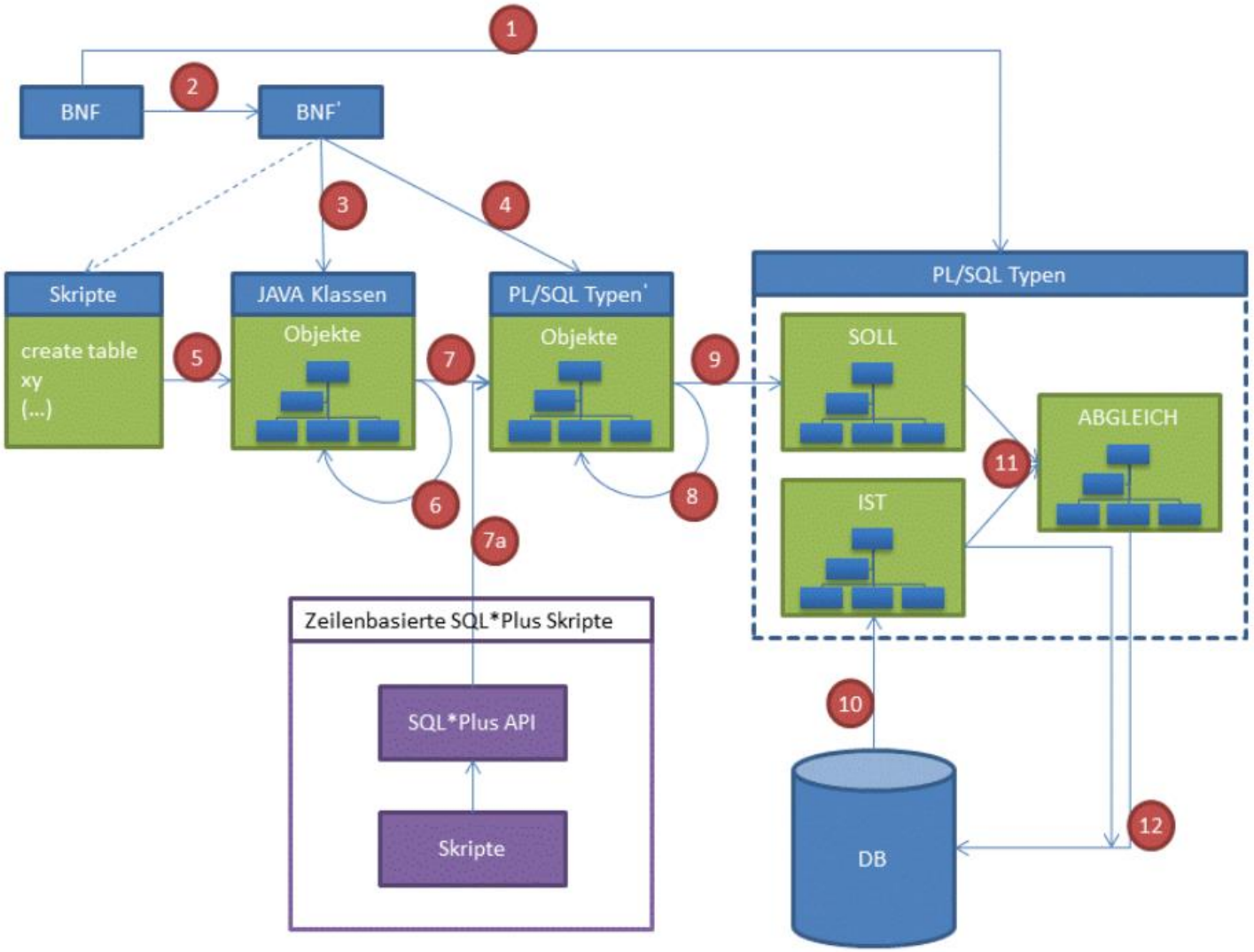
Tabellenskript in Orcas

```
create table order_items
(
  orit_id   number(15)                          not null,
  version   number(15)       default "0"        not null,
  ordr_id   number(15)                          not null,
  item_id   number(15)                          not null,
  price     number(8,2)                         not null,
  text      varchar(50),
  quantity  number(4)                           not null,


  constraint orit_pk primary key (orit_id),
  constraint orit_pricecheck check ("price>0"),
  index      orit_text_upper_ix (upper(text)),
  constraint orit_uc unique (ordr_id, item_id) disabled,
  index      orit_version_ix (version),
  constraint orit_item_fk foreign key (item_id) references items (item_id),
  constraint orit_ordr_fk foreign key (ordr_id) references orders (ordr_id) on delete cascade

  comment on table is "Ausführliche Beschreibung von Order_Items";
  comment on column version is "Ausführliche Beschreibung der Spalte Order_Items.Version";
);
```

Orcas - Architektur

- SQL-ähnliche Syntax auf Basis von xtext
- Abgleich mit dem Data Dictionary durch PL/SQL-Packages
- Separates Orcas – Schema (optional)
- Delta-Skripte werden mit SQL*Plus ausgeführt
- Aufruf von Orcas über Ant
- Verwendet intern Gradle
- Syntax-, Java- und PL/SQL-Extensions
- Reverse Engineering

Funktionsweise



3 Orcas Extensions

Orcas Extensions

- Syntax Extensions
- Domain Extensions
- Java-Extensions
- PL/SQL-Extensions
- PL/SQL-Reverse-Extensions

Syntax - Extensions

```
package de.opitzconsulting.orcas.syntax_extensions;

public class AliasSyntaxExtension extends BaseSyntaxExtension
{
  public void run()
  {
    addField( new FieldReference( "Table", "name" ), new NewFieldDataIdentifier( "alias", false ) );
  }
}
```

```
create table orders alias ordr
(
  orderdate         date                not null,
  tracking_number   varchar2(20)        not null,
  shipping_country  varchar2(2)         not null
);
```

Java Extensions

```
package de.opitzconsulting.orcas.extensions;

import de.opitzconsulting.orcasDsl.*;

public class CharColumn extends TableVisitorExtension
{
  @Override
  protected void handleTable( Table pTable )
  {
    for( Column lColumn : pTable.getColumns() )
    {
      if( lColumn.getData_type() == DataType.VARCHAR2 )
      {
        lColumn.setByteorchar( CharType.CHAR );
      }
    }
  }
}
```

Domain - Extensions

```
create table tab_a
(
  tab_a_id          number(10)          not null,
  somevalue         varchar2(100)
);

create table tab_b
(
  tab_b_id          number(10)          not null,
  somevalue         varchar2(100)
);
```

Domain - Extensions

```
define table domain id_table
(
  add column column-name(table-name||"_"||column-name) ( id number(10) not null)
);
```

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

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

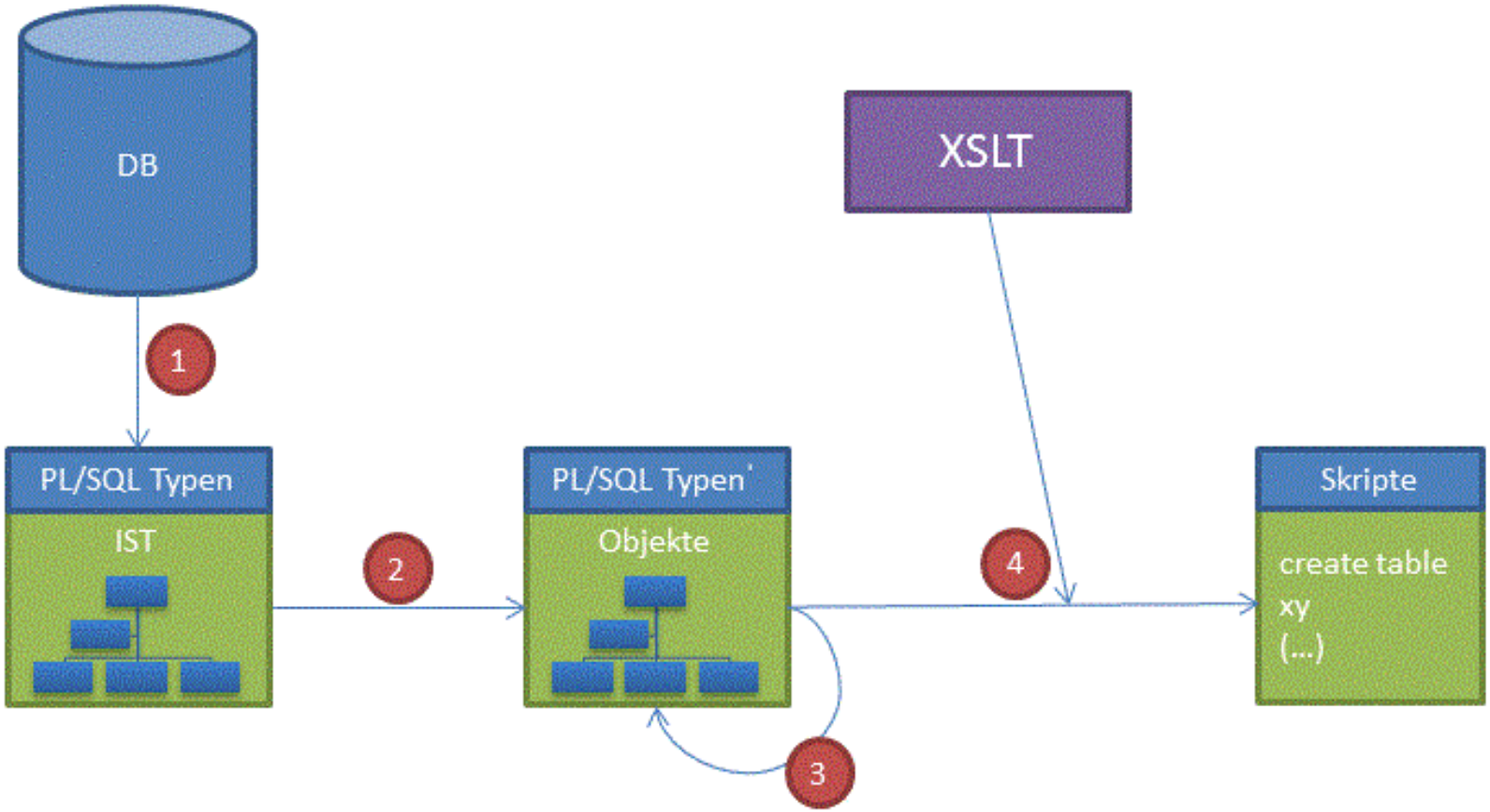
Vorhandene Domain Extensions

- **Table Domains**
  - Spalten hinzufügen
  - History-Tabelle hinzufügen
  - Historisierungstrigger generieren
- **Column Domains**
  - Datentyp, Precision/Scale
  - Not-null
  - Defaultwerte
  - Constraints erzeugen (z. B. PK mit Sequenz)

PL/SQL Extensions

```
create or replace package body pa_char_column is
  procedure handle_table( p_syex_table in out nocopy ot_syex_table )
  is
    v_syex_column ot_syex_column;
  begin
   for i in 1..p_syex_table.i_columns.count
    loop
      v_syex_column := p_syex_table.i_columns(i);

      if( ot_syex_datatype.is_equal( v_syex_column.i_data_type, ot_syex_datatype.c_varchar2 ) = 1)
      then
        v_syex_column.i_byteorchar := ot_syex_chartype.c_char;
      end if;

       p_syex_table.i_columns(i) := v_syex_column;
    end loop;
  end;

  function run( p_input in ot_syex_model ) return ot_syex_model
  is
```

Reverse Engineering



4 Arbeiten mit Orcas

Orcas – Getting started

- **Java ab Version 6**
- **Ant (+ ant-contrib-1.0b3.jar)**
- **Gradle**
- **Oracle Client**
- **Installationsanleitung: http://opitzconsulting.github.io/orcas/docs/installation/**

Klassifikation von DB-Objekten

- **Static**
  - Table (Partitionen, Materialized View logs)
  - Materialized View
  - Sequence
- **Replaceable**
  - Package
  - Function/Procedure
  - View
  - Trigger

Orcas Ant Tasks

- orcas_install: Einrichtung des Orcas DB-Schemas
- orcas_execute_statics: Abgleich statischer DB-Objekte
- orcas_drop_replaceables: Löschen von replaceable Objects
- orcas_execute_script: Ausführung eines SQL*Plus-Skripts
- orcas_execute_scripts: Ausführung von SQL*Plus-Skripten
- orcas_execute_one_time_scripts: Ausführung von Einmalskripten (Migrationsskripten)
- orcas_extract: Erstellt Orcas-Skripte des aktuellen Schemas (Reverse Engineering)

Ant Builddatei

```
<project name="database">
  <property name="orcas_dir" value="../orcas_core"/>
  <import file="${orcas_dir}/orcas_default_tasks.xml"/>

  <property name="orcas.default_user" value="${username_schemaowner}"/>
  <property name="orcas.default_password" value="${password_schemaowner}"/>
  <property name="orcas.default_user_orcas" value="${username_orcas}"/>
  <property name="orcas.default_password_orcas" value="${password_orcas}"/>
  <property name="orcas.default_tnsname" value="${database}"/>

  <target name="orcas_install">
    <orcas_install user="${username_dba}" password="${password_dba}"/>
  </target>

  <target name="build_all" depends="">
    <orcas_initialize extensionfolder=
       "${distributiondir}/../../orcas/orcas_extensions"/>
    <orcas_execute_statics
      scriptfolder="tables"
      dropmode="${dropmode}"
      logname="statics"/>
    <orcas_drop_replaceables
      logname="dropreplaceables"/>
    <orcas_execute_scripts
      scriptfolder="views"
      logname="views" />
  </target>
</project>
```

Einmalskripte

- Einmalig pro Zielschema auszuführende Migrationsskripte
- Datenmigration (Post-Skripte)
- Pre-Skripte: Beispiel Column-Rename
- Alternativ: Liquibase-Integration

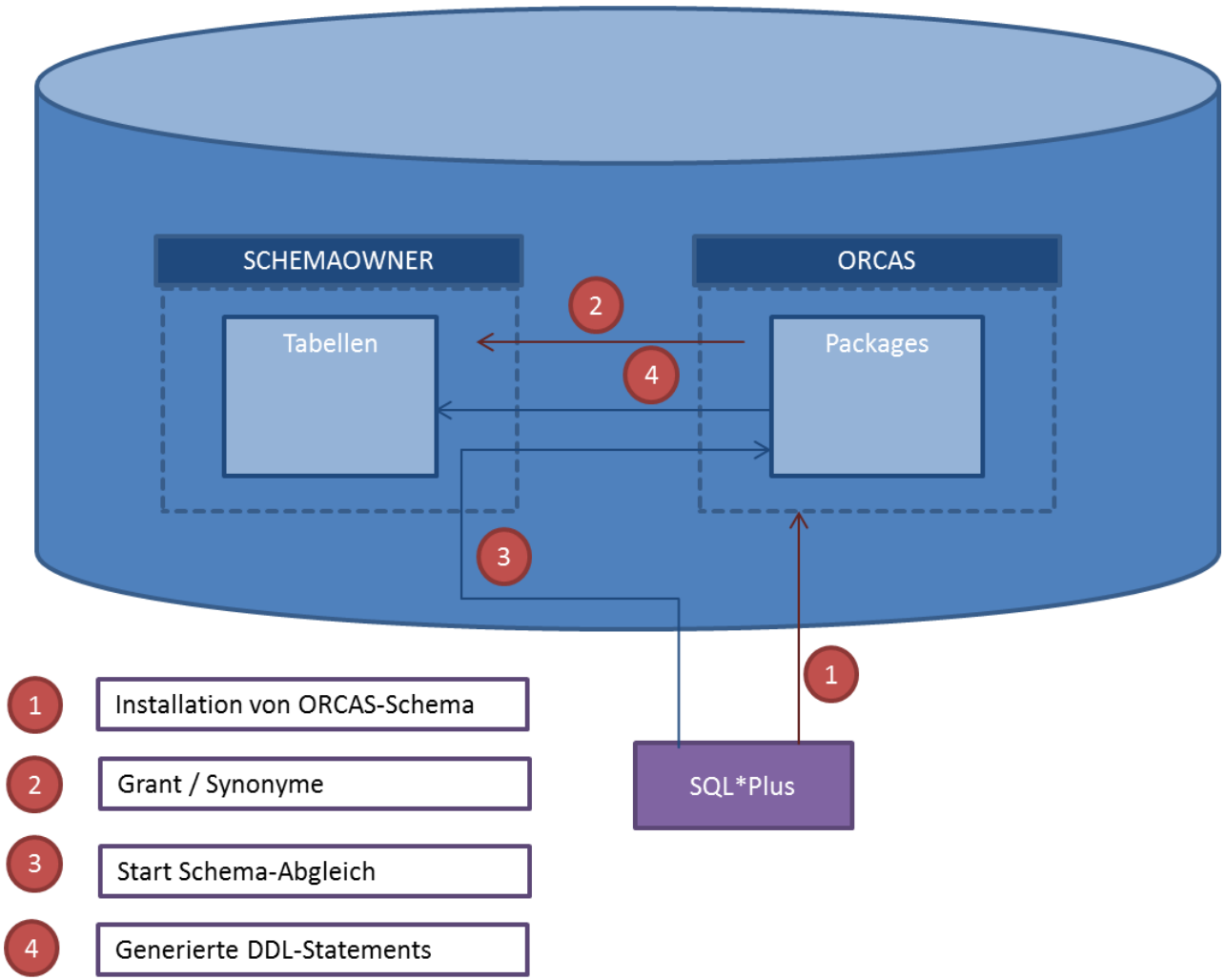
Beispiel für ein PL/SQL-Projekt

- Pre-Skripte
- Tabellen
- Package Header
- Functions, Procedures
- Views
- Package Bodies
- Trigger
- Post-Skripte

Genereller Aufbau

- Separates Orcas-Schema
- Skripte selbst sind Schema-neutral
- Ein Build-Aufruf pro Schema
- Orcas-Schema wird an Schemaowner gegranted
- Deployment mit Schemaowner
- Aufruf von Orcas mit ‚authid current user‘

Schemaabgleich mit Orcas



5 Einsatzszenarien

Entwicklungsprojekte

- Umfangreiche Änderungen an Code und Tabellen
- Downtime unkritisch
- Direkte Ausführung der Änderungen
- Löschen und Neuinstallation aller Replaceables

Deployment mit minimaler Downtime

- **Schemavergleich ohne direkte Skriptausführung**
  - Schemavergleich bei laufender Produktion
  - Ausführung der Delta-Skripte separat
  - Lösung für Deltaermittlung bei Replaceables
- **Trennung von Tabellen und Replaceables**
  - Separate Schemata für Tabellen und Replaceables
  - Aufbau eines Schemas für die neue Replaceable-Version
  - Umschalten auf neues Schema nach Deployment

Einsatz für Schemavergleich

- **Export der Statics (Tabellen, Materialized Views, Sequenzen) aus dem Quell-Schema (Reverse Engineering)**
- **Unterstützung für Export von Replaceables (types, views, packages, triggers, functions, procedures)**
- **Deployment auf Zielumgebung wie in anderen Szenarien**

Fazit

- Orcas ermöglicht Continuous Delivery der Datenbank
- DB-Skripte verhalten sich wie Anwendungssourcen
- Deployment unabhängig vom Ist-Zustand der DB
- Flexibel erweiterbar
- Reverse Engineering
- Open Source (Apache License)
- Einschränkung: nur für Oracle RDBMS

Weitere Informationen

- **Continuous Integration**
  http://www.thoughtworks.com/continuous-integration
- **Database Refactoring**
  http://databaserefactoring.com/
- **Orcas auf Github:**
  https://github.com/opitzconsulting/orcas/
- **Dokumentation:**
  http://opitzconsulting.github.io/orcas/de/
- **Whitepaper/Blogeinträge:**
  http://opitzconsulting.github.io/orcas/docs/talking-about/

Question & Answer

Discussion Area in Jazz 2

You don't have to put an abrupt stop to exciting discussions at the end of a talk: In Room "Jazz 2" you can keep conversation going.

Kontakt

Dr. Olaf Jessensky
Senior Consultant

OPITZ CONSULTING Deutschland GmbH
olaf.jessensky@opitz-consulting.com

www.opitz-consulting.com

youtube.com/opitzconsulting

@OC_WIRE

slideshare.net/opitzconsulting

xing.com/net/opitzconsulting