

# Daten für Performance-Tests

C1 SetCon Performance Day  
5. Oktober 2010 13:00 - 13:45

Volker Bergmann

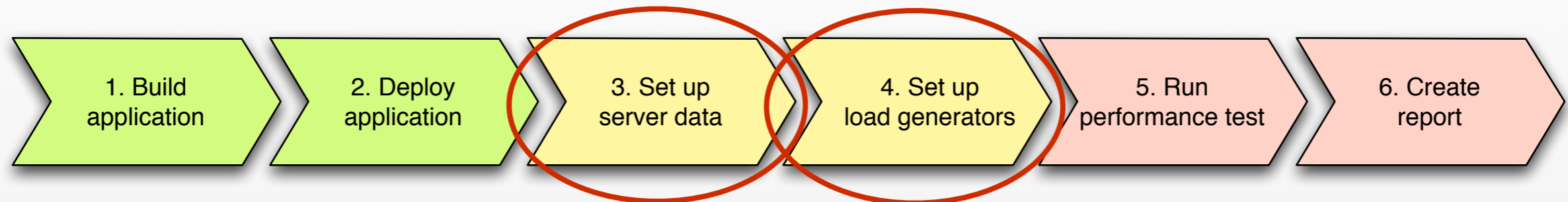
# Kennen Sie den?

- 13 Jahre professionelle Software-Entwicklung
- zahlreiche Großprojekte
- Analyse, Design, Implementation, Test
- Bestrebt, Software-Qualität schon in frühen Projektphasen zu sichern
- Performance-Fokus, besonders for J2EE Server-Software
- Entwicklung von Open Source Testwerkzeugen
- Er heißt Volker Bergmann



# Performance Testing Process

- Diese Schritte sind Bestandteil fast jeden Performance-Tests



- Datenaspekte (Schritte 3 und 4)
  - oft vernachlässigt
  - essentiell für die Aussagekraft des Tests
  - Thema des Vortrags

# Performanz-Prognose

- Eine möglichst genaue Prognose der Produktionsperformance erfordert den Test mit
  - produktionsähnlicher Hardware
  - produktionsähnlichem Client-Verhalten
  - produktionsähnlichen Daten
    - Menge
    - Struktur
    - **aber: bessere Qualität**

# Produktionsdaten in Performance-Tests

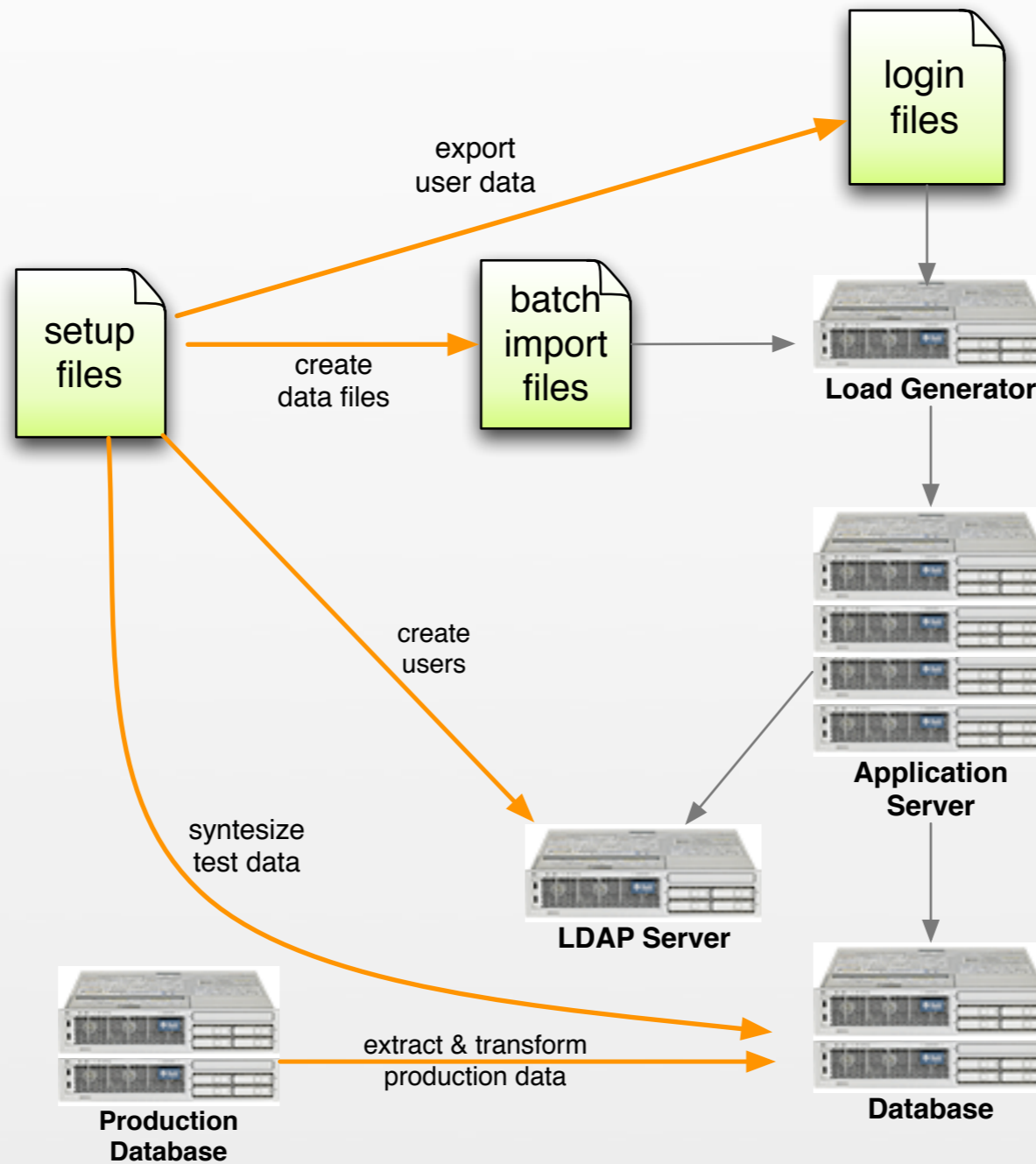
- **Sind Produktionsdaten**
  - **verfügbar?** (Outsourcing, erste Version, sensible Daten, Urlaub, Migration, ...)
  - **anonymisiert?** (Sensitivität der Daten, Kompetenz der Verantwortlichen, Meldepflichten...)
  - **valide?** (10-30% fehlerhafte Altdaten üblich durch Abbruch von Prozessen, Abschalten von Constraint-Validierung bei Migration, ...)
  - **anwendbar?** (obsoletere Daten, neuer Release erfordert neue Daten, geändertes Nutzerverhalten, mehr Nutzer in der Zukunft...)

# Produktionsdaten-Fazit

- Produktionsdaten sind der perfekte Test-Maßstab  
...für die Vergangenheit
- Im Regelfall müssen zusätzlich zur Extraktion von Produktionsdaten...
  - Datenvalidität geprüft werden
  - Untermengen extrahiert werden
  - Daten hinzugeneriert werden
- Extraktion und Anonymisierung von Produktionsdaten stellen nur einen Teil der Datenaufbereitung für Performance-Tests dar

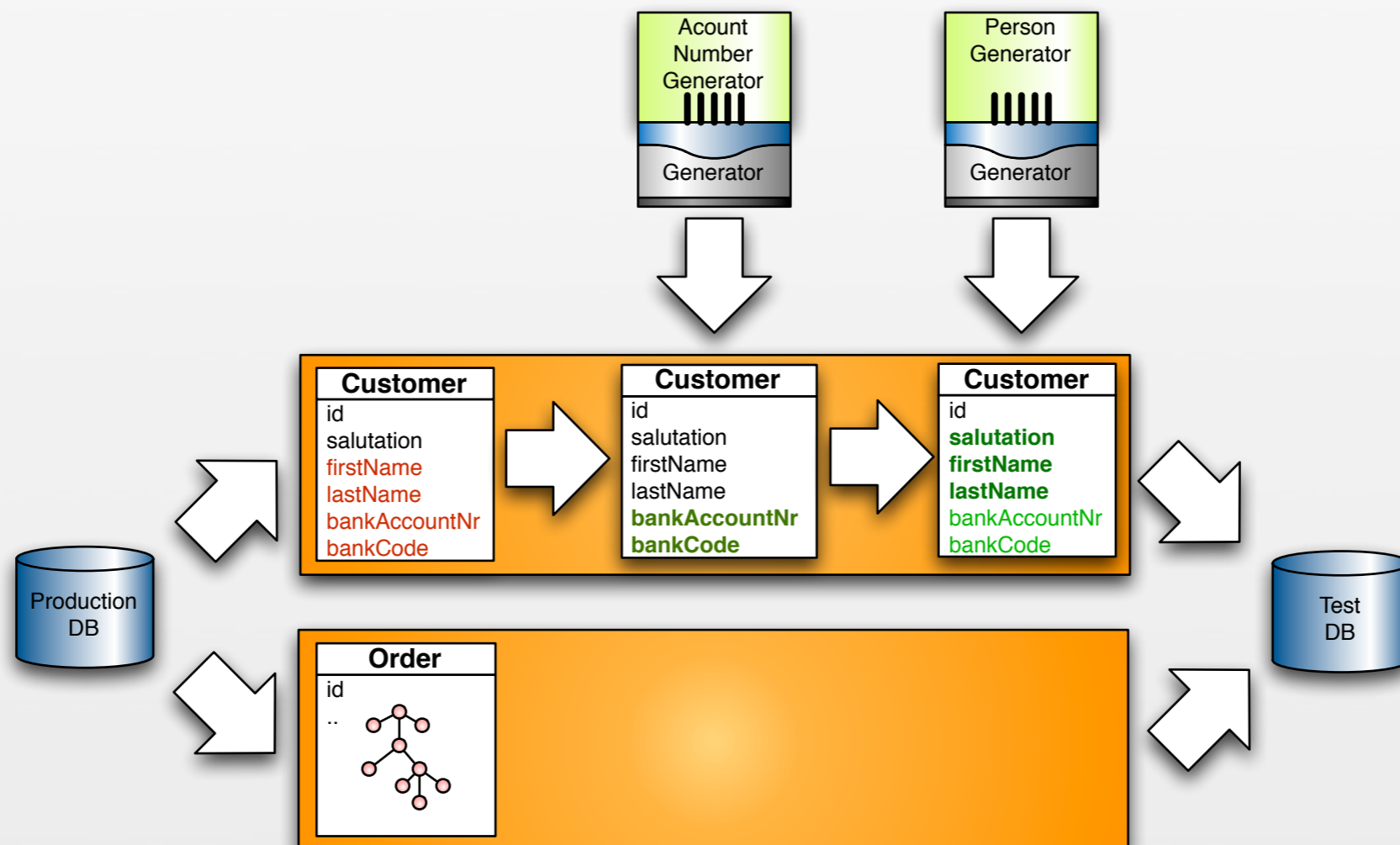


# Datenaspekte im Performance-Test



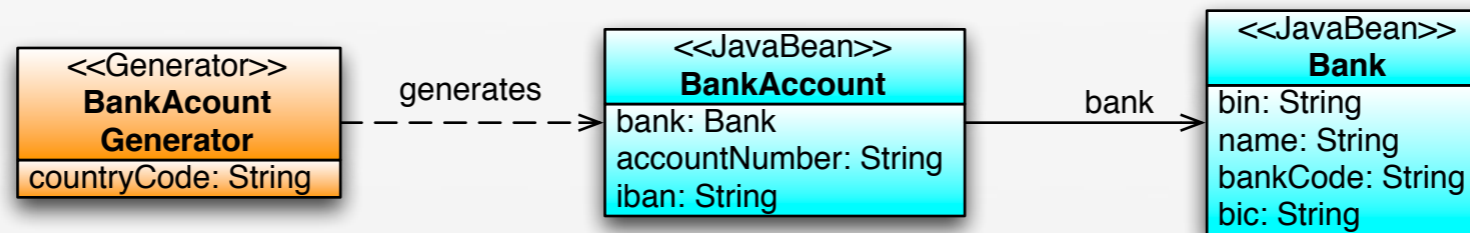
# Extraktion & Anonymisierung

- Oft können Datenstrukturen 1:1 in das Testsystem übertragen werden
- Nur vertrauliche Daten müssen anonymisiert werden



# Anonymisierung mit Benerator

- Generatormodule erzeugen Objekte oder konsistente Objektgraphen, deren Komponenten auf das Zieldatenmodell abgebildet werden



```

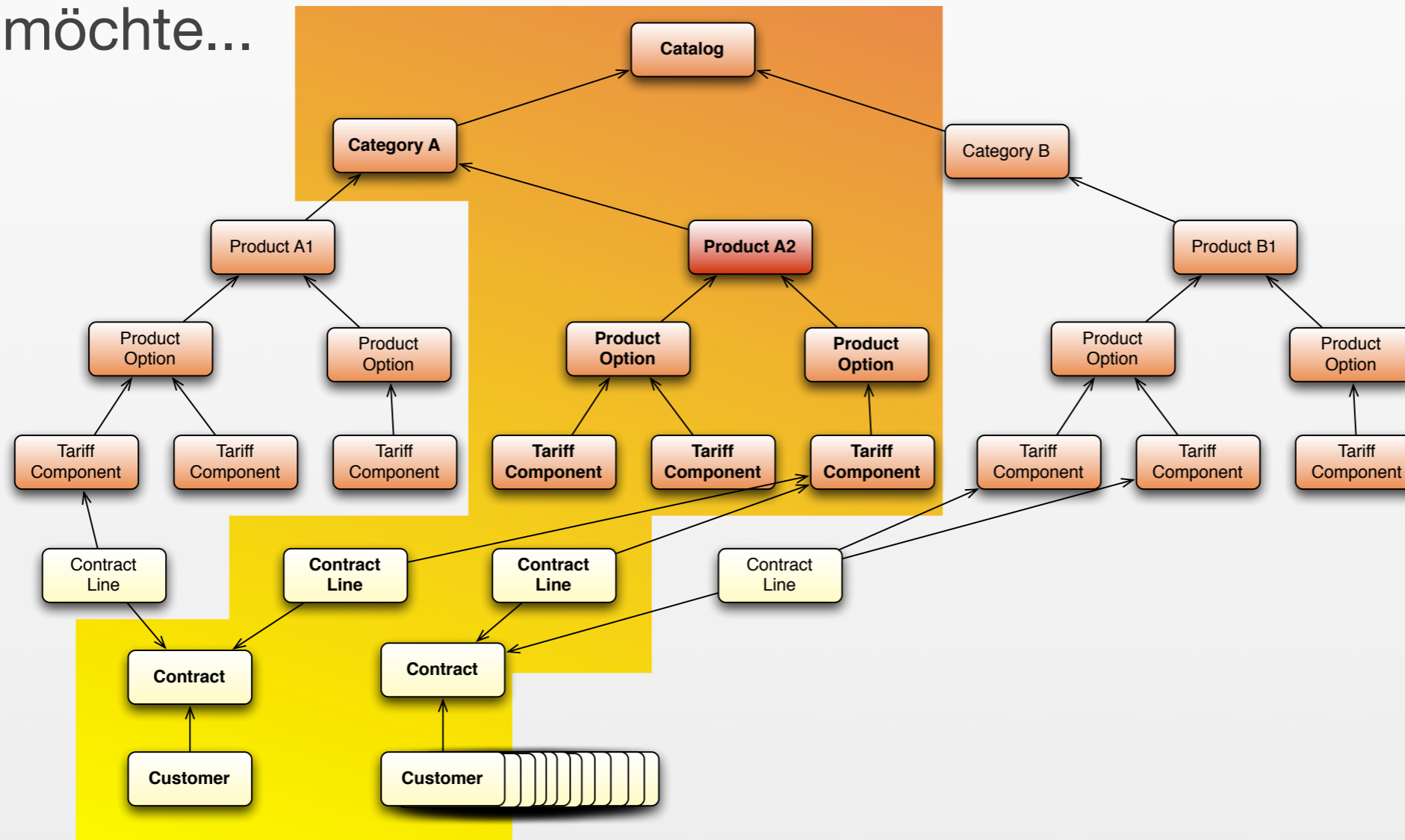
<database id="prod" readOnly="true" ... />
<database id="testdb" ... />

<iterate source="prod" type="CUSTOMER" consumer="testdb">
  <variable name="acct" generator="BankAccountGenerator" />
  <attribute name="BANK_ACCOUNT_NR" script="acct.accountNumber" />
  <attribute name="BANK_CODE" script="acct.bank.bankCode" />
</iterate>

<iterate source="prod" type="ORDER" consumer="testdb" />
  
```

# Datenbank-Subsetting

- Wenn man z.B. den Produktkatalog auf ‚Product A2‘ beschränken möchte...



- ...wie wahrt man dann referentielle Integrität?

# Jailer Jailer

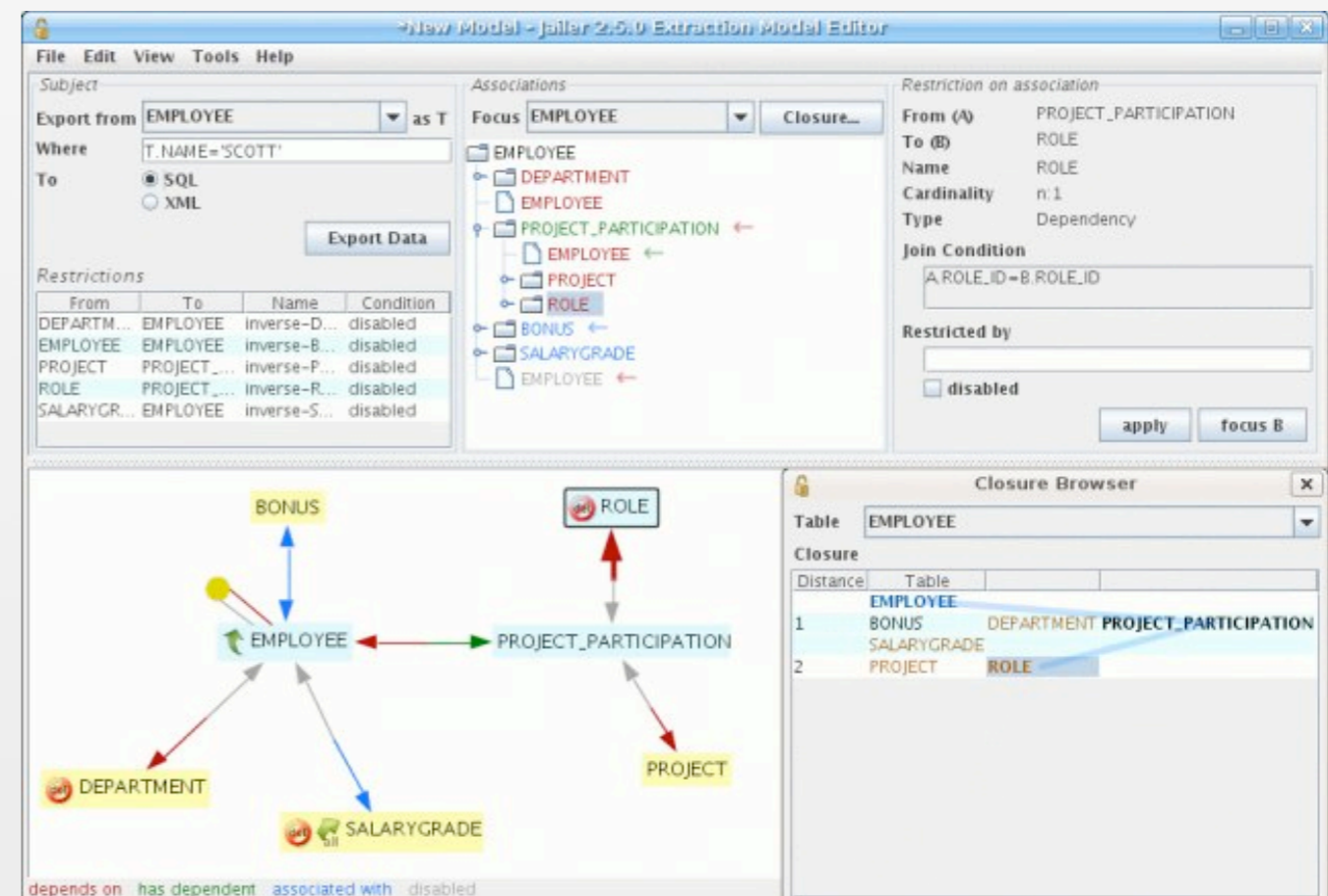
- Datenbank-Subsetting
- Open Source, aktives Projekt, Aktuelle Version: 3.4.7 (2010)
- Erkennt explizite Foreign Key-Beziehungen in der Datenbank

- Konfiguration:

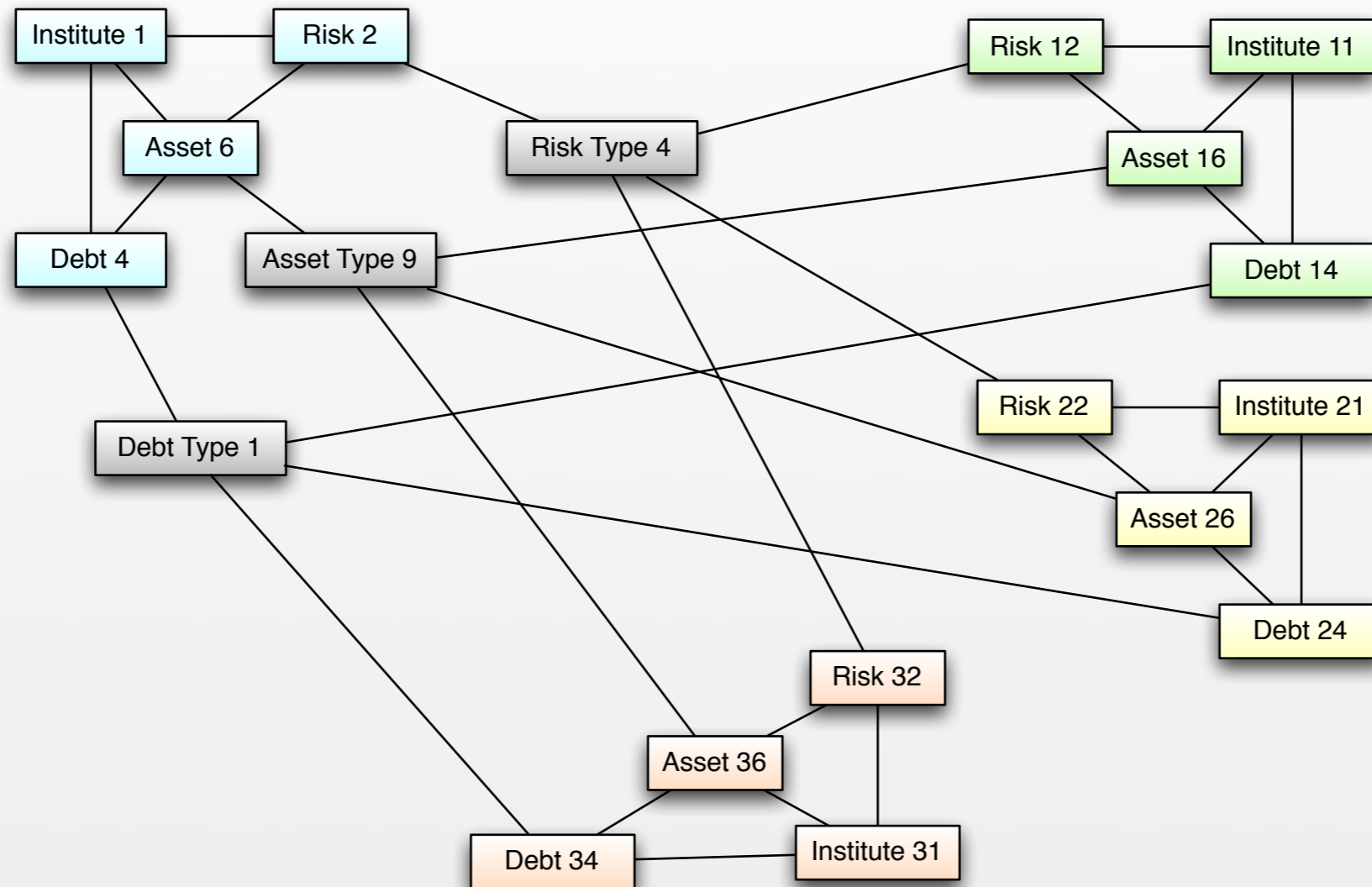
- implizite Foreign Keys
- Aggregation/Referenz

- Export in

- SQL
- XML
- DbUnit

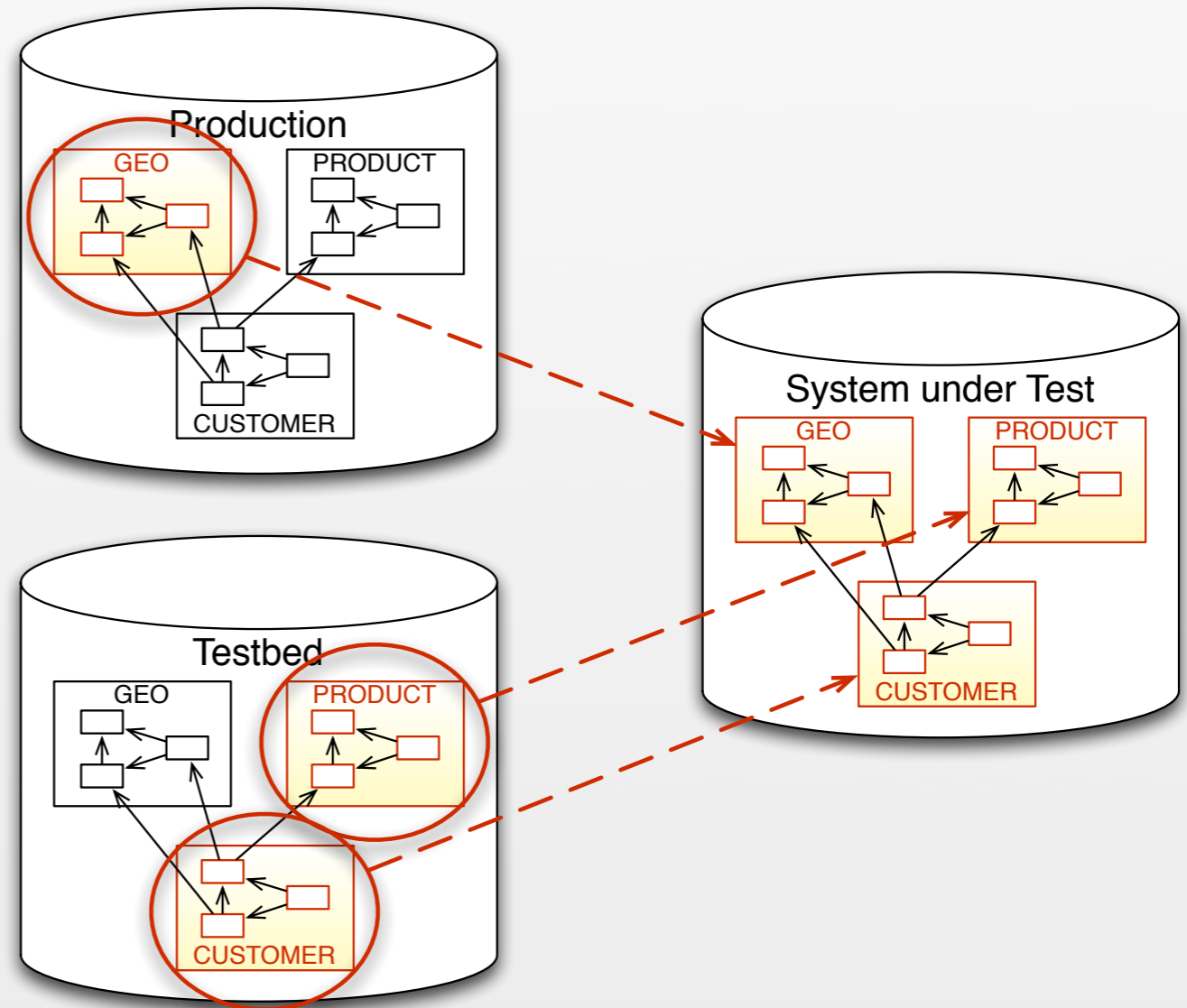


# Datenreplikation



# Zusammenführen von Datenquellen

- Hauptaufgaben:
  - Vergabe neuer Primärschlüssel
  - Ersetzen von Referenzen
  - Konsistenzprüfung
- Wird bislang durch kein Standardwerkzeug unterstützt
- Geplant für Benerator



# Datengenerierung

- Synthetische Erzeugung von Daten
  - ETL-Funktionalität: Extraktion und Transformation von Daten aus Datenbanken oder Dateien
  - Generator-Funktionalität: Generierung von Zufallsdaten / zufälligen Kombinationen von Basisdaten
- Kernproblem: Explizite und implizite Bedingungen in
  - Datenbank
  - Applikation
- Für die meisten Performance-Tests müssen zumindest Teile der Daten generiert werden
- Erfahrung: Typischerweise unbrauchbare Zufallsdaten

# Wenn man gute Daten generieren könnte...

- **Test von Entwicklungsszenarien:**
  - Mobilfunk-Beispiel: Wie wäre die Systemperformance in einem Jahr, wenn wir durch uns unseren neuen Flatrate-Tarif 30% Neukunden gewinnen und alle Kunden 50% länger telefonieren würden?
- **Test auf Spezialprobleme:**
  - Beispiel Adressvalidierung: Gibt es bei der automatischen Korrektur von Postadressen Spezialkonstellationen, die problematisch sind („Killeradressen“?)
  - Ihr Beispiel: Der typische Prio 1-Performance-Incident nach 2 Monaten Produktion
  - Lösungsansatz: Generative Abdeckung aller Kombinations-/ Fehlermöglichkeiten (deckt u.U. auch funktionale Fehler auf)

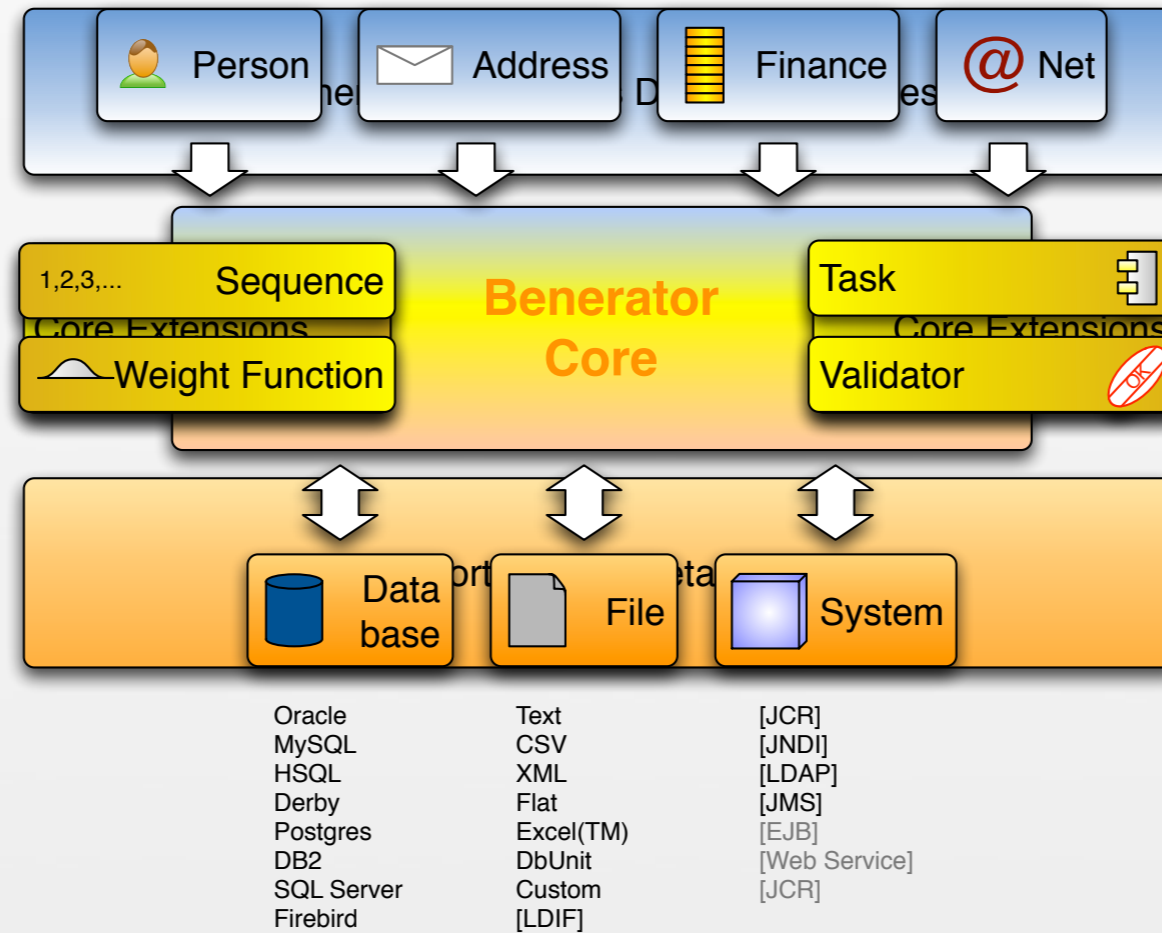
# Wenn man gute Daten generieren könnte...

- **Performance-Test in frühen Projektphasen**
  - Start mit ‚Smoke‘-Daten, die sehr schnell konfiguriert sind
  - Verfeinerung der Generierung bei Implementationsfortschritt
- **Continuous Performance Testing**
- **Show-Case-Erstellung** und Schulungsdaten als Nebenprodukt

# Das kann man: Benerator...

- entstand, um für Performance Tests Produktionsdaten simulieren zu können
- hat zum Hauptziel, valide Daten in großer Menge zu erzeugen und Produktionsdaten zu anonymisieren
- erlaubt die Generierung extrem komplexer Strukturen
- ist Open Source
- ist unabhängig vom Betriebssystem
- ist unabhängig von der Zielplattform, RDBMS, XML, CSV, XLS, ...
- erlaubt die Definition fachlicher Generierungskomponenten
- ist das Standardtool zur Testdatengenerierung

# Benerator-Architektur



# Benerator-Anwender



bislang 10.000 Downloads

# Datengenerierung mit Benerator

Definiere eine Datenbank ,db' -->

```
<database id="db" ... />
```

Führe DDL/SQL-Skripte aus -->

```
<execute uri="create_tables.sql" target="db" />
```

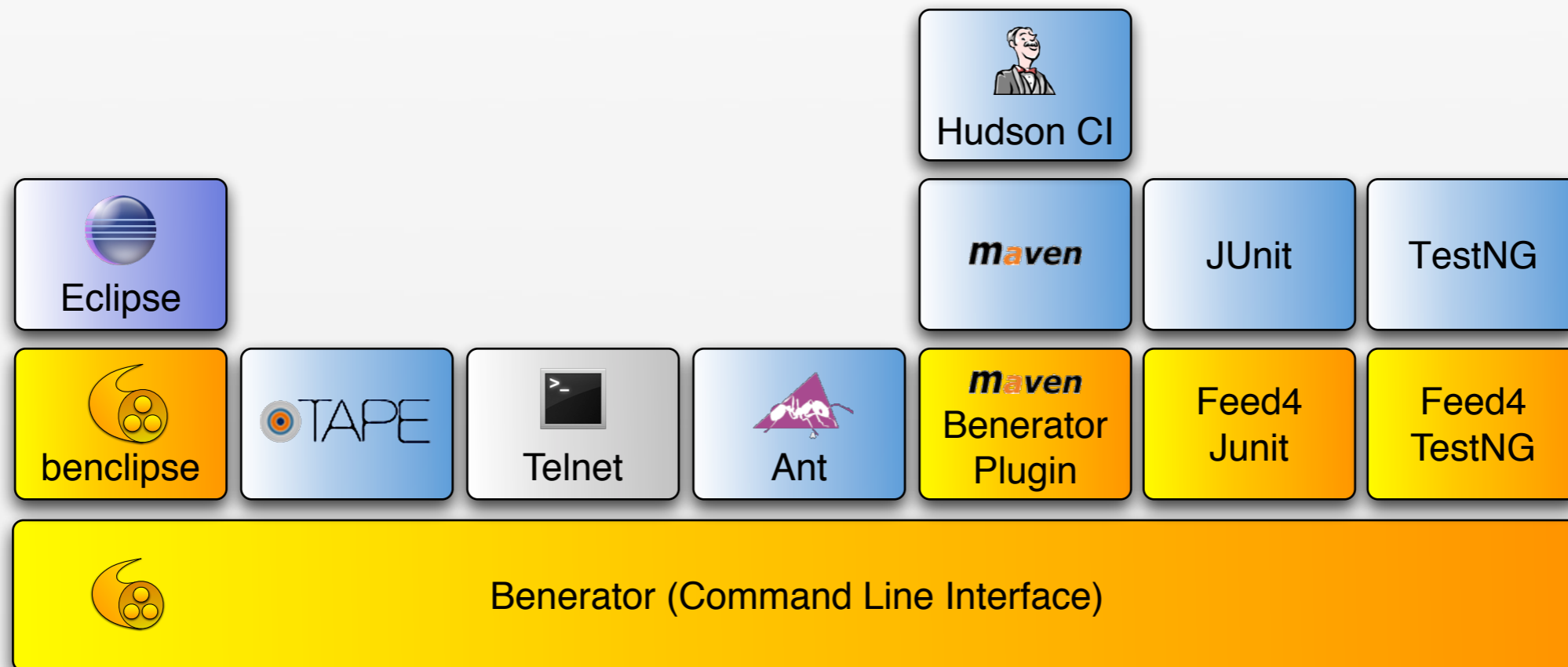
Importiere Daten aus DbUnit-Dateien -->

```
<iterate source="products.dbunit.xml" consumer="db" />
```

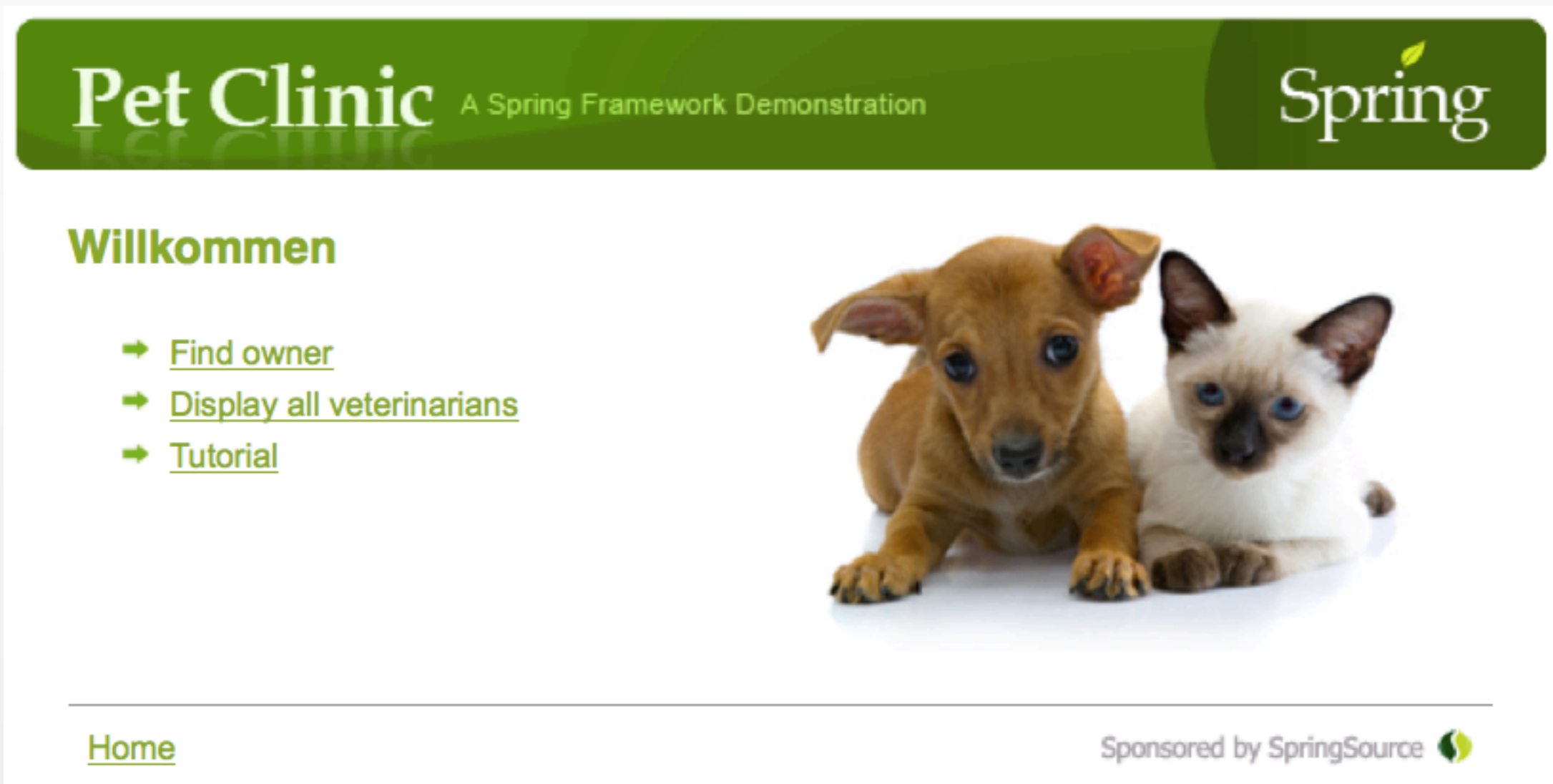
Generiere valide Daten -->

```
<generate type="db_order" consumer="db">  
  <id name="id"  
    generator="new DBSequenceGenerator('SEQ_ORDER',db)"/>  
  <reference name="customer" targetType="db_customer"  
    distribution="random" />  
  <attribute name="created_at"  
    generator="CurrentDateGenerator" />  
  <attribute name="created_by" script="this.customer" />  
</generate>
```

# Benerator-Anbindungen


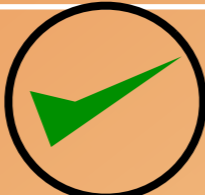





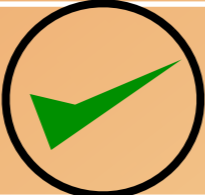


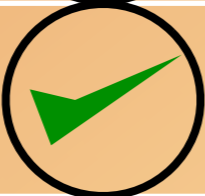










# Demo: Spring Pet Clinic



The screenshot shows the Spring Pet Clinic application interface. At the top, there is a green header bar with the text "Pet Clinic A Spring Framework Demonstration" on the left and the "Spring" logo on the right. Below the header, the word "Willkommen" is displayed in green. To the left of the main content area, there is a list of three menu items, each preceded by a green arrow: "Find owner", "Display all veterinarians", and "Tutorial". To the right of the menu items is a photograph of two small animals: a brown puppy and a white Siamese kitten. At the bottom left, there is a "Home" link. At the bottom right, there is a "Sponsored by SpringSource" logo.

# Datenbezogene Werkzeuge

	Benerator	 Jailer	Altova MapForce
Anonymization			
Datenbank Subsetting			
Replikation			
Datengenerierung			
Zusammenführung	 (geplant)		
*x OS			

# Danke für Ihre Aufmerksamkeit



Volker Bergmann  
volker@databene.org

...noch Fragen?