Verteidigung Diplom

UML/OCL für die Integritätssicherung in Datenbankanwendungen

Sten Löcher

Technische Universität Dresden
Fakultät Informatik
Lehrstuhl Softwaretechnologie

Inhalt des Vortrages

- Aufgabenstellung
- Konzept für die Integritätssicherung
- UML‘99-Musterkatalog
- Werkzeugunterstützung
- Zusammenfassung der Ergebnisse
- Ausblick

I. Aufgabenstellung

Grundgedanken:

- Objektorientierte Modellierung persistenter Daten einschließlich Integritätsbedingungen mit Hilfe von UML und OCL .
- Grundlage für die Speicherung der persistenten Daten sind relationale Datenbanksysteme.
- Überprüfung der mit OCL formulierten Integritätsbedingungen mit Hilfe von Mechanismen der Datenbank.

Voraussetzungen:

- Am Institut entwickelte Technologie für die Verarbeitung von OCL-Ausdrücken und die Übersetzung in SQL:
  - OCL-Compiler
  - Musterkatalog für die Übersetzung OCL in SQL
  - SQL-Codegenerator
- Einschränkung der Anwendbarkeit der entwickelten Technologien durch gängige relationale Datenbankmanagementsysteme.

Aufgabe:

- Entwicklung einer UML/OCL-basierten Strategie für die Integritätssicherung in relationalen Datenbanken.
- Beachtung der technologischen Einschränkungen durch die Datenbanksysteme.
- Entwurf der Werkzeugunterstützung für die Strategie bzw. das Konzept.

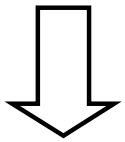
II. Konzept für die Integritätssicherung

Analyse der Anforderungen

- Wie sollen die Geschäftsregeln beschrieben werden ?
  - zugrundeliegendes Sprachprinzip und Notation
  - **Voraussetzung für diese Arbeit:** OCL
- Wo sollen die Geschäftsregeln abgelegt werden ?
  - Implementation in der Anwendung, in der Datenbank oder in einer separaten Software
  - **Voraussetzung für diese Arbeit:** Datenbank

- Wann sollen die Geschäftsregeln überprüft werden ?
  - kontinuierlich bzw. sofort nach der Manipulation von Daten oder nur zu bestimmten Zeitpunkten
- Wer ist für die Evaluierung der Geschäftsregeln verantwortlich ?
  - Anwendung, Datenbank oder separate Software

- Wann sollen die Geschäftsregeln überprüft werden ?
  - kontinuierlich bzw. sofort nach der Manipulation von Daten oder nur zu bestimmten Zeitpunkten
- Wer ist für die Evaluierung der Geschäftsregeln verantwortlich ?
  - Anwendung, Datenbank oder separate Software



flexibler Ansatz notwendig

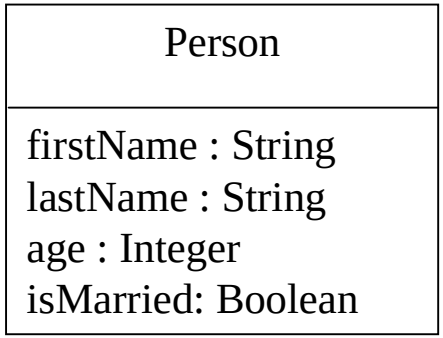
Integritäts-View-Konzept:

Ein Integritäts-View ist ein aus einer Integritäts-bedingung generierter View, der auf Anfrage alle Tupel einer Tabelle selektiert, die gegen die formulierte Integritätsbedingung verstoßen.

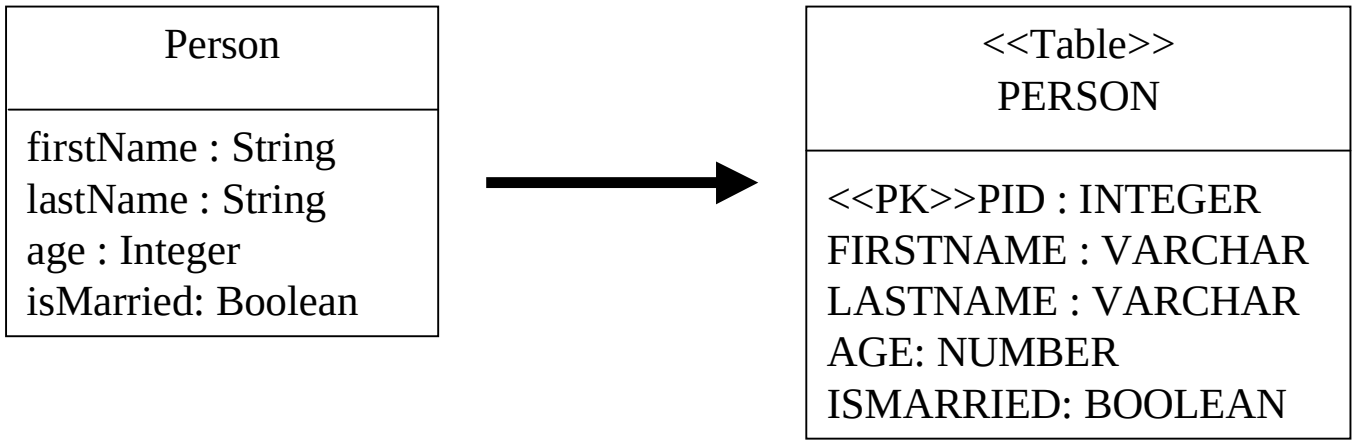
Bsp. für die Verwendung der Integritäts-Views:

- anwendungsgesteuerte Evaluation
- SQL92-Assertion-Replacement
- ECA-Trigger-Templates

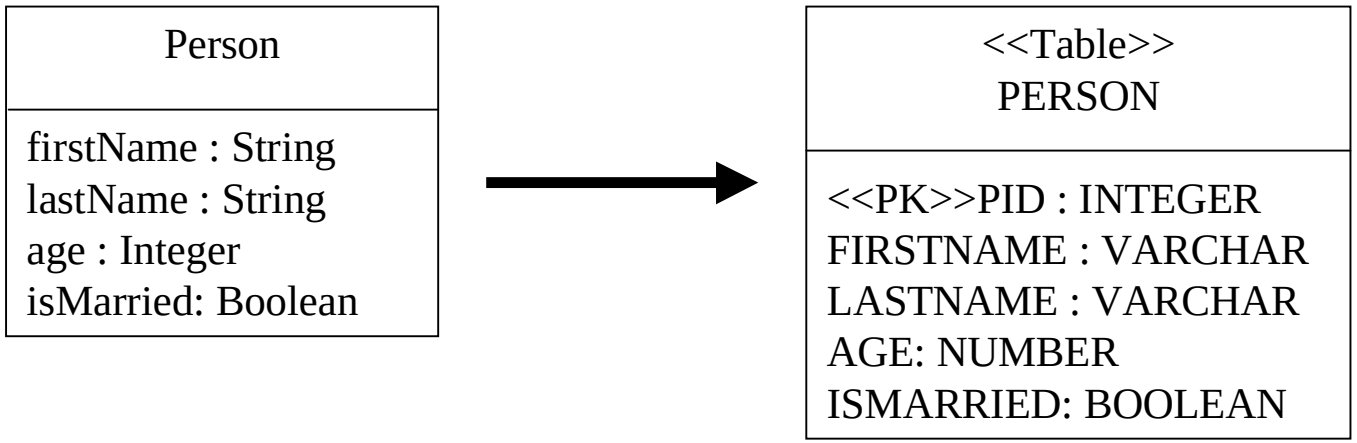
Beispiel:


Person
firstName : String
lastName : String
age : Integer
isMarried: Boolean

Beispiel:

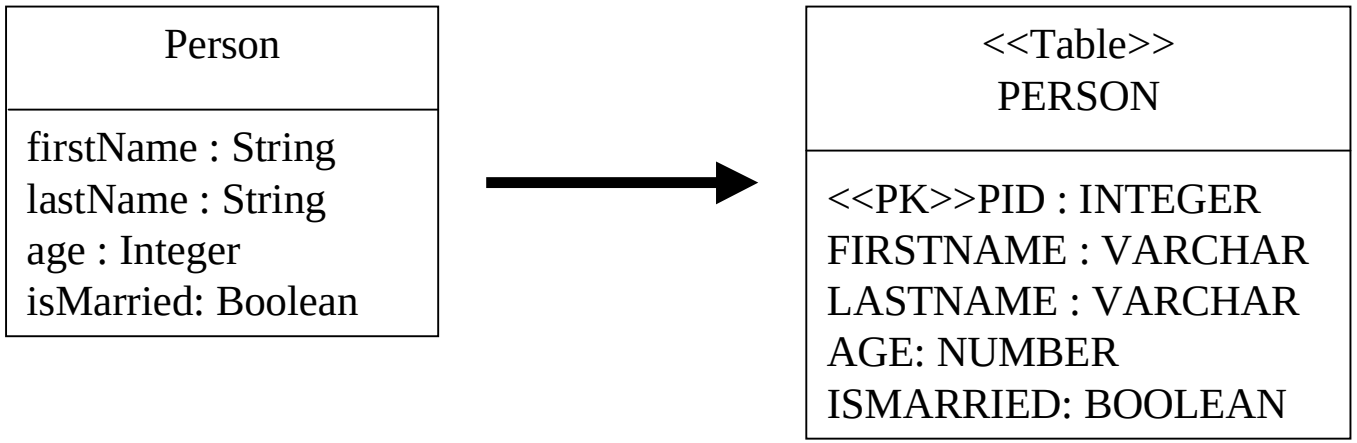

Person
firstName : String
lastName : String
age : Integer
isMarried: Boolean
<<Table>>
PERSON
<<PK>>PID : INTEGER
FIRSTNAME : VARCHAR
LASTNAME : VARCHAR
AGE: NUMBER
ISMARRIED: BOOLEAN

Beispiel:



context Person inv ageOfMarriage:
(isMarried = true) implies (age>=18)

Beispiel:



context Person inv ageOfMarriage:
(isMarried = true) implies (age>=18)

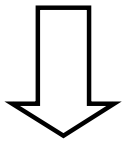
```
create view AGEOFMARRIAGE as
(select * from PERSON
 where not (not (isMarried = true)
                or (age>=18)))
```

III. UML‘99 Musterkatalog

- beschreibt die Übersetzung von OCL-Ausdrücken in SQL
- besteht aus acht Mustern, wobei jedes Muster die Übersetzung für ein OCL-Sprachkonzept beschreibt
- intensive Arbeit mit dem Katalog während GB und DA

III. UML‘99 Musterkatalog

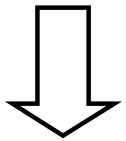
- beschreibt die Übersetzung von OCL-Ausdrücken in SQL
- besteht aus acht Mustern, wobei jedes Muster die Übersetzung für ein OCL-Sprachkonzept beschreibt
- intensive Arbeit mit dem Katalog während GB und DA



Probleme und Kritikpunkte

Probleme und Lösungen:

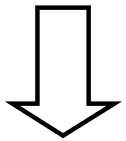
- keine Berücksichtigung verschiedener objektrelationaler Abbildungsstrategien



Objekt-View-Konzept zur Vereinfachung der Abbildung von Ausdrücken über Vererbungsstrukturen + Empfehlungen für bestimmte Abbildungsvarianten

Probleme und Lösungen :

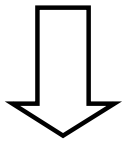
(2) unzureichend geklärte Abbildung des Zugriffs auf Daten und Metadaten des Anwendungsmodells



neues Muster MODEL TYPE QUERY zur Beschreibung der Abbildung von Operatoren über Typen

Probleme und Lösungen:

(3) keine Betrachtung der Besonderheiten bei der Abbildung von Ausdrücken mit Sequenzen

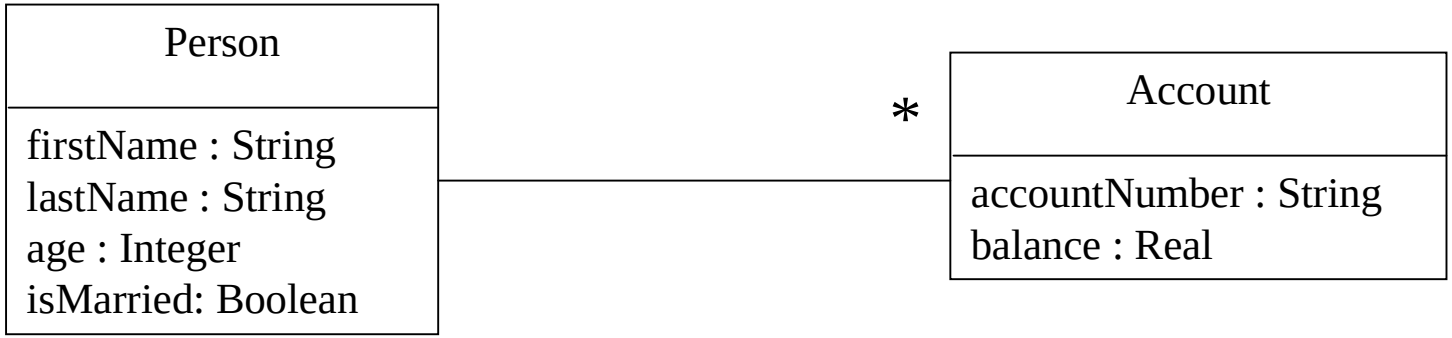


neues Muster SEQUENCE für die Beschreibung der Abbildung von Ausdrücken mit Sequenzen

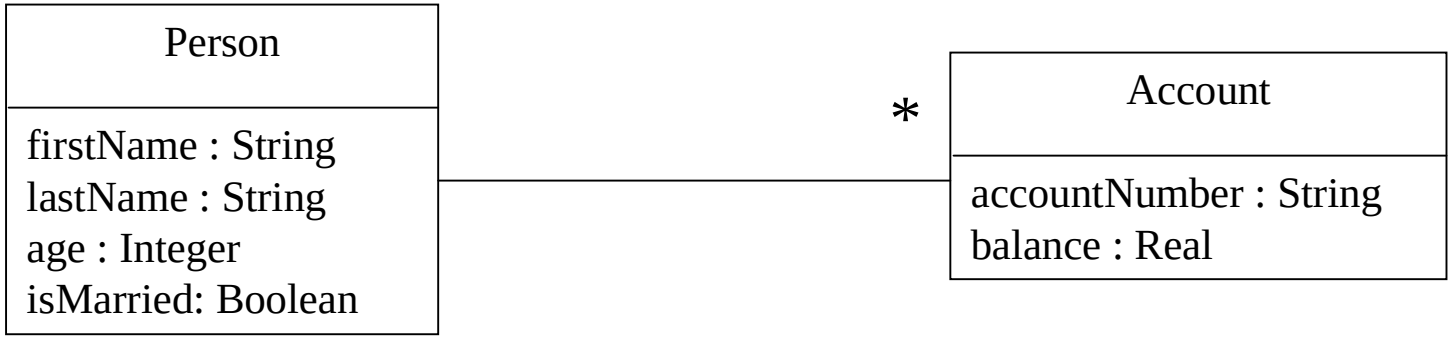
Erfahrungen mit Oracle:

- SQL92-Standard unzureichend implementiert
- dadurch Probleme bei der Anwendung der Muster
- Hauptproblem: abgeleitete Tabellen in der FROM-Klausel von SELECT-Statements werden nur eingeschränkt unterstützt
- Alternative: Umformung der Abbildungsmuster unter Zuhilfenahme von Mengenoperatoren (UNION, UNION ALL, MINUS, INTERSECT)
- Aber: Einschränkungen bei Operationen über Bags und Sequenzen

Beispiel:

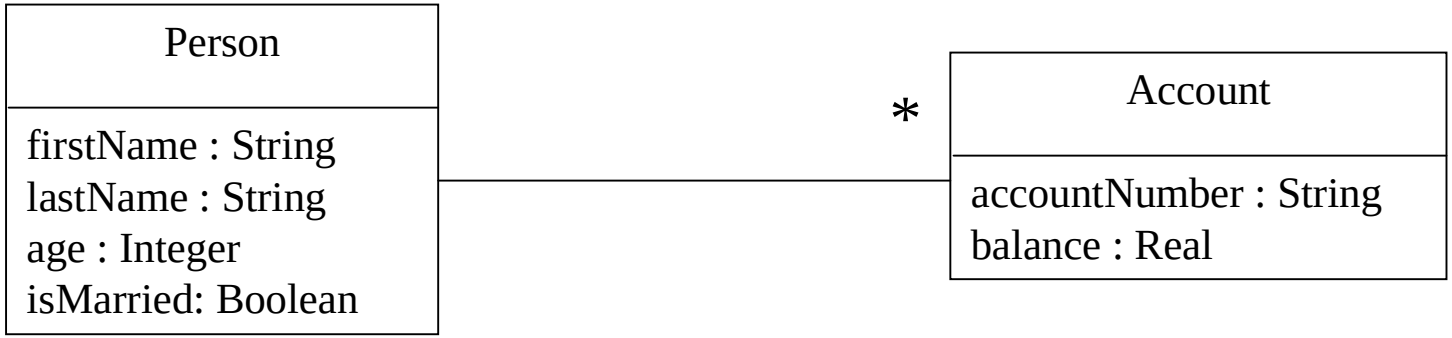

Person
firstName : String
lastName : String
age : Integer
isMarried: Boolean
*
Account
accountNumber : String
balance : Real

Beispiel:



context Person inv minBal:
self.account->forAll(balance > 0)

Beispiel:




```
context Person inv minBal:
self.account->forAll(balance > 0)
```

```
create view MINBAL as
(select * from PERSON SELF
 where not(
   not exists (
        select PID
        from (select PID from ACCOUNT
                where PID = SELF.PID)
        where PID in (
                select PID from PERSON
                where not (balance > 0)))
   )
)
```

Beispiel:

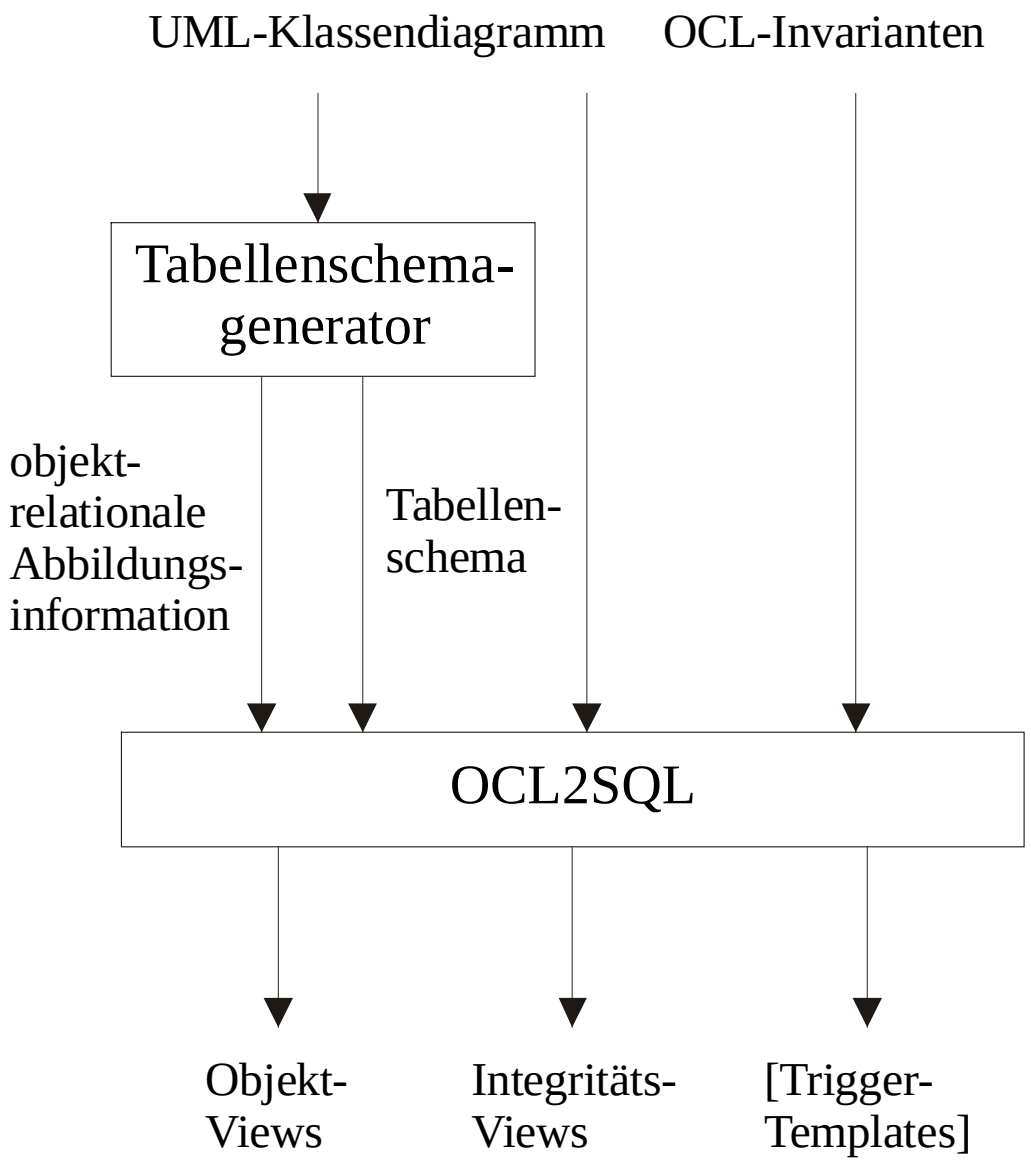
```
create view MINBAL as
(select * from PERSON SELF
 where not(
    not exists (
        select PID
        from (select PID from ACCOUNT
                where PID = SELF.PID)
        where PID in (
                select PID from PERSON
                where not (balance > 0)))
    )
)
```

Beispiel:

```
create view MINBAL as
(select * from PERSON SELF
 where not(
    not exists (
          (select PID from ACCOUNT
           where PID = SELF.PID)
          minus
          (select PID from PERSON
           where (balance > 0))
    )
)
```

IV. Werkzeugunterstützung

- für die entwickelten Konzepte sollte eine Werkzeugunterstützung entworfen und implementiert werden
- basierend auf dem Dresden OCL Toolkit
  - OCL-Compiler
  - XMI-Schnittstelle für Typinformationen
  - OCL-Editor
  - SQL-Codegenerator-Prototyp


UML-Klassendiagramm
OCL-Invarianten
Tabellenschema-
generator
objekt-
relationale
Abbildungs-
information
Tabellen-
schema
OCL2SQL
Objekt-
Views
Integritäts-
Views
[Trigger-
Templates]

Komponenten des OCL2SQL-Tools:

- Objekt-View-Generator
  - benutzt objektrelationale Abbildungsinformationen
  - erzeugt für jede Klasse einen Objekt-View
- Integritäts-View-Generator
  - SQL-Codegenerator aus GB
  - Integration Objekt-View-Konzept
  - Überarbeitung XML-Musterkatalog
  - Registrierung verwendete Tabellen für Trigger-Generator
  - gesonderte Behandlung Boolescher Werte
- Trigger-Generator
  - erzeugt Assertion-Ersatztrigger und ECA-Templates

V. Zusammenfassung der Ergebnisse

Ergebnisse:

- Konzept für die Integritätssicherung
- Überarbeitung des Musterkataloges
- Vorschlag für Objekt-View-Konzept
- Entwurf und Implementation der Werkzeugunterstützung
- Implementation eines Schemagenerator-Prototypen
- Test des Werkzeuges an einer Beispielanwendung

Erfahrungen/Schlußfolgerungen:

- UML/OCL prinzipiell im DB-Design einsetzbar
- Struktur des UML‘99-Musterkatalog richtig
- wichtige Rolle der objektrelationalen Abbildung
- Objekt-Views vereinfachen Übersetzungsprozeß
- direkte Verwendung der Muster nicht möglich
- Alternativen ermöglichen eingeschränkte Anwendung
- Tabellenschemata-Generatoren in CASE-Tools nicht zufriedenstellend

VI. Ausblick

Was kann noch getan werden:

- SQL-Codegenerierung
- Effizienz und Praxisrelevanz
- objektrelationale Abbildung
- Weiterentwicklung des Objekt-View-Konzeptes
- Erzeugung von standardkonformen SQL-Code