Prof. Dr.(PL) Michael Unterstein
Fachhochschule Frankfurt/Main

Logischer Entwurf und Dokumentation objektrelationaler Oracle Datenbanken

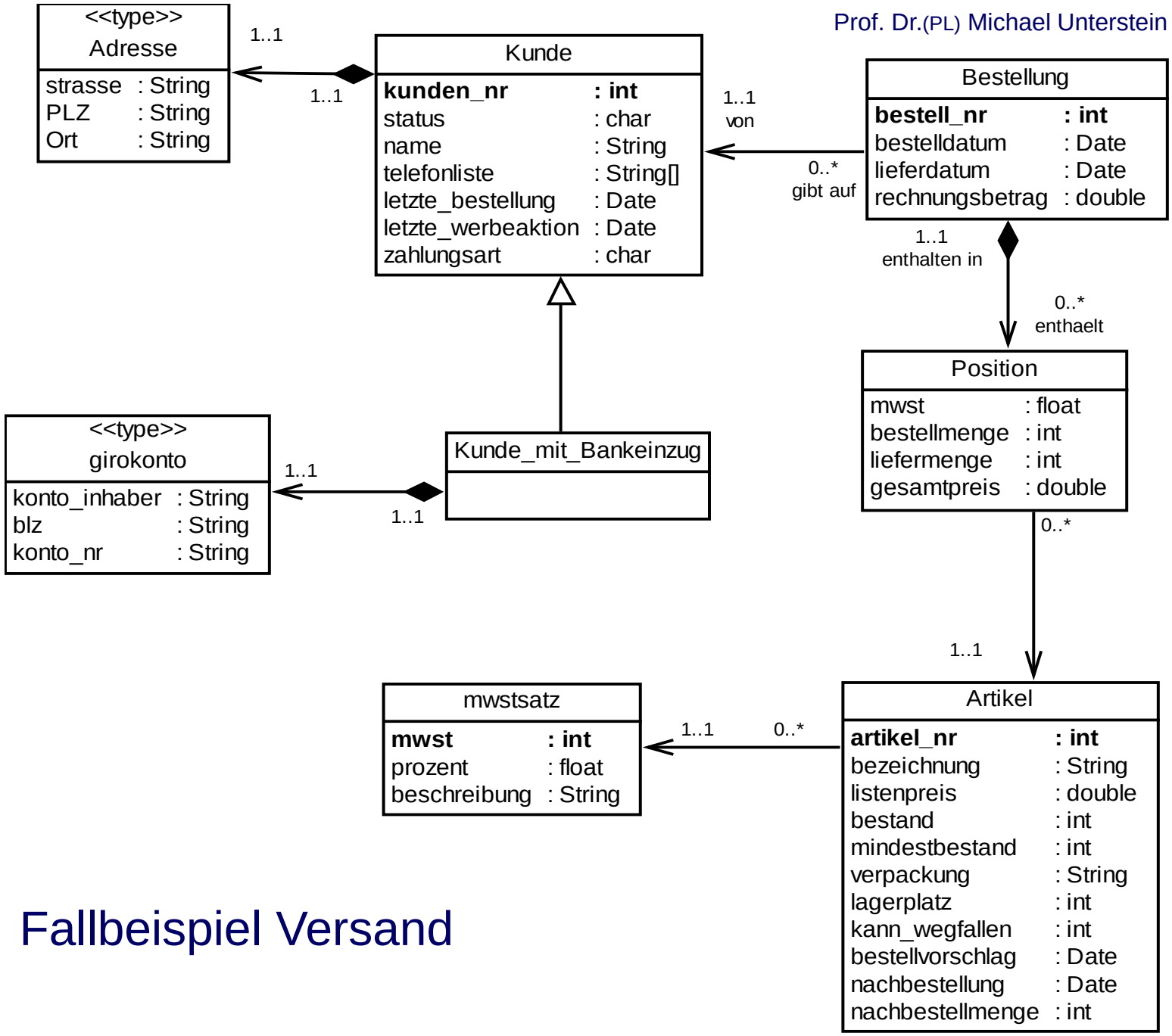
Agenda

1. Einleitung und Motivation
2. Fallbeispiel
3. Probleme mit UML Diagrammen
4. Beispielhafte Umsetzung des Entwurfs
5. Plädoyer und Vorschlag für ein eigenes Datentypendiagramm
6. Dokumentation einer OR-Datenbank mithilfe des Systemkatalogs

Einleitung und Motivation

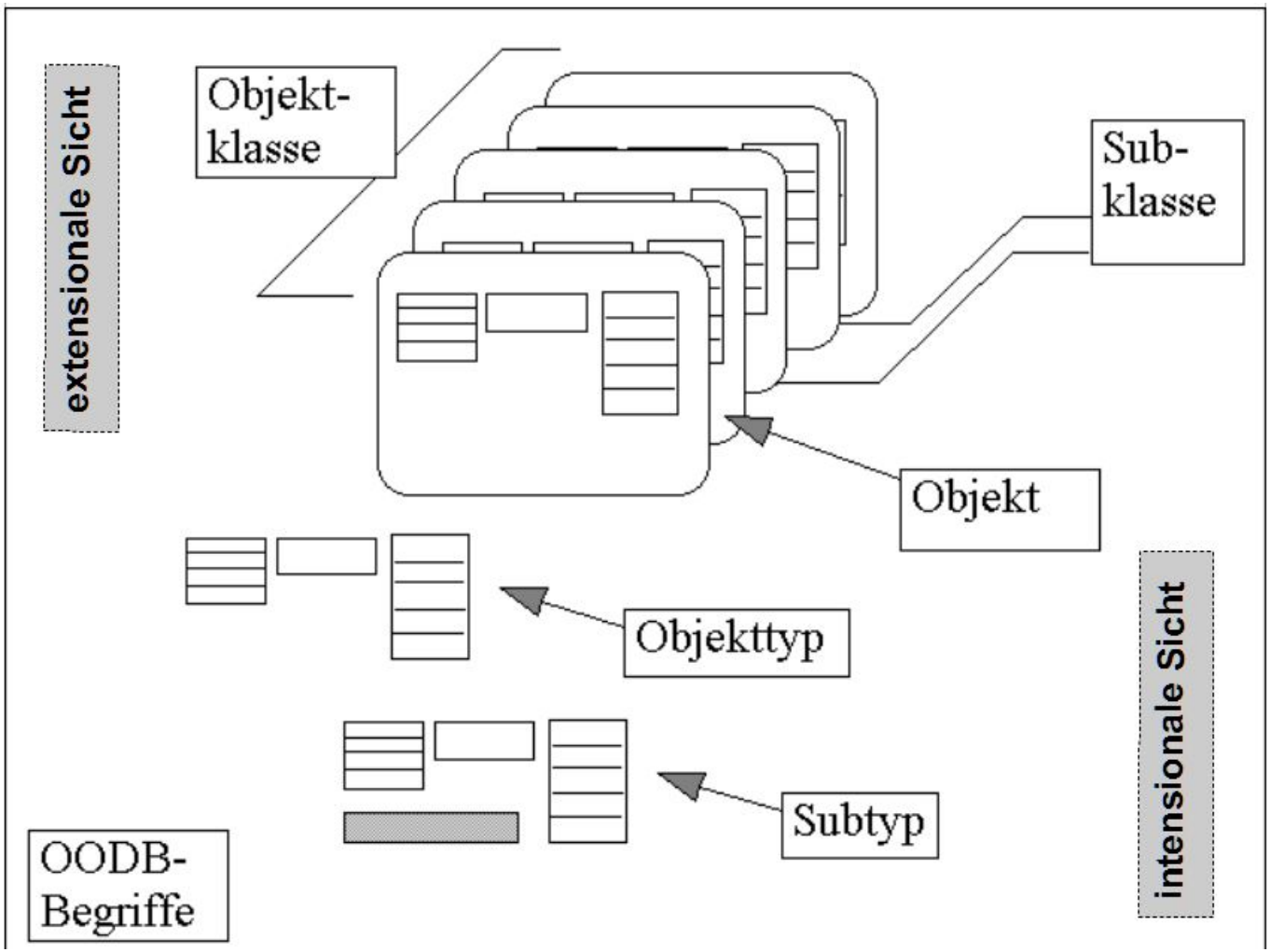
- Logischer Entwurf und Dokumentation objektrelationaler Datenbanken stellen besondere Anforderungen
  - Vielzahl benutzerdefinierter Datentypen
  - mehrstufige Abhängigkeiten
- UML Klassendiagramme
  - sind für den konzeptionellen Entwurf gedacht
  - der darauf aufbauende logische Entwurf enthält diverse Freiheiten des Entwicklers
  - Hierfür gibt es keine Diagrammkonvention

Prof. Dr.(PL) Michael Unterstein



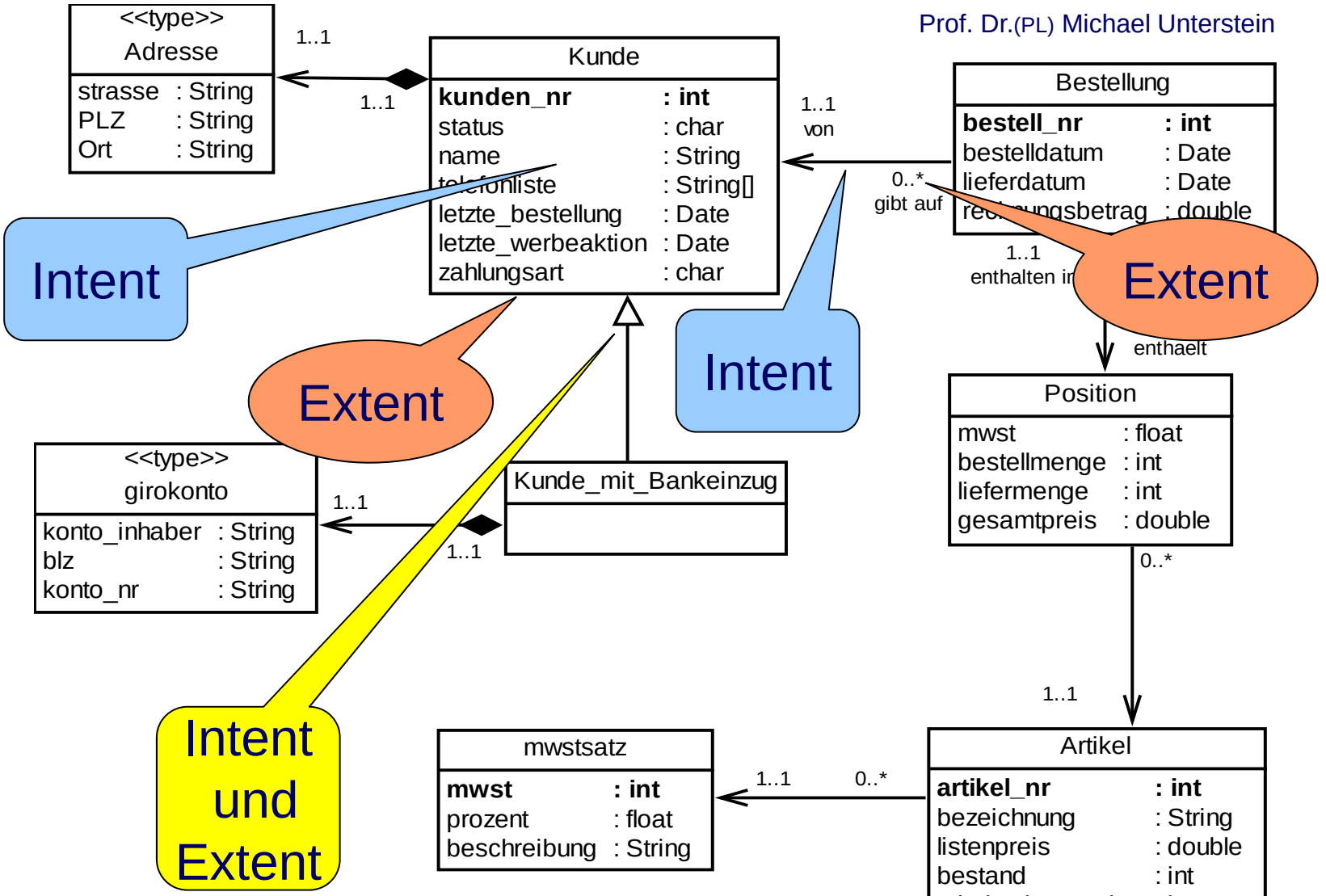
Fallbeispiel Versand

Intensionale und extensionale Sicht



... werden im Klassendiagramm vermischt

Prof. Dr.(PL) Michael Unterstein



Extent und Intent am Fallbeispiel Versand

... im objektrelationalen Datenmodell getrennt

```
-- Intent, ein Typ:
CREATE OR REPLACE TYPE otyp_kunde AS OBJECT (
    kunden_nr                 number (4,0),
    status                    varchar2(1),
    name                      varchar2(30),
    telefonliste          varr_telefonliste,
    adresse             otyp_adresse,
    letzte_bestellung         date,
    letzte_werbeaktion        date,
    zahlungsart               char(1)
)
    NOT FINAL; -- Subtypen erlaubt
/
-- Extent, eine typisierte Tabelle:
CREATE TABLE otab_kunde OF otyp_kunde
       SUBSTITUTABLE AT ALL LEVELS
       (CONSTRAINT pk_kunde
        PRIMARY KEY (kunden_nr)
        -- weitere Constraints auf Extentebene);
```

verschiedene Möglichkeiten, die Beziehungen zu realisieren

- Ein Objekt direkt in ein anderes einbetten, indem wir es als Datentyp für ein Attribut verwenden.
- Ein Objekt für sich bestehen lassen und von anderen Objekten mit einer Referenz darauf verweisen (Typkonstruktor REF)
- relationale Fremdschlüssel benutzen, was aber dem objektorientierten Paradigma widerspräche
- Mehrere Objekte über eingebettete Tabelle
- Mehrere Objekte über VARRAY einbetten
- ... Kombinationen dieser Möglichkeiten VARRAY von REF ...

```
CREATE OR REPLACE TYPE otyp_adresse AS OBJECT (
    strasse         varchar2(30),
    plz             varchar2(5),
    ort             varchar2(30),
);

-- Anlage eines Typs fuer bis zu 5 Telefonnummern
CREATE OR REPLACE TYPE varr_telefonliste AS VARRAY (5) OF
VARCHAR2(20);
/

-- Anlage eines Typs fuer Kunden
CREATE OR REPLACE TYPE otyp_kunde AS OBJECT
(
-- Zuvor definierte Typen werden benutzt.
-- Keine Constraints moeglich
    kunden_nr               number (4,0),
    status                  varchar2(1),
    name                    varchar2(30),
    telefonliste       varr_telefonliste,
    adresse             otyp_adresse,
    letzte_bestellung       date,
    letzte_werbeaktion      date,
    zahlungsart             char(1))
    NOT FINAL
```

Beispielhafte Umsetzung Teil1

```
CREATE OR REPLACE TYPE otyp_girokonto AS OBJECT
   (konto_inhaber varchar2(30),
    blz           varchar2(8),
    kontonr       varchar2(10)
);
/

-- Anlage des Subtyps für Kunden
-- mit Bankeinzug

CREATE OR REPLACE TYPE otyp_kunde_mit_bankeinzug UNDER otyp_kunde
(
  bankverbindung otyp_girokonto
);
/
```

Beispielhafte Umsetzung Teil2

```
CREATE OR REPLACE TYPE otyp_mwstsatz AS OBJECT (
        mwst  integer,
        prozent  number (3,3),
        beschreibung  varchar2(10)
        );

CREATE OR REPLACE TYPE otyp_artikel AS OBJECT
     (artikel_nr       varchar2(4),
      mwst             REF otyp_mwstsatz,
      bezeichnung      varchar2(15),
      listenpreis      number(8,2),
      ...)

CREATE OR REPLACE TYPE otyp_position AS OBJECT (
      pos_artikel  REF otyp_artikel,
      mwst         number (4,3),
      -- tatsaechlich angewandter
      -- MWST-Satz als Dezimalzahl
      bestellmenge   number (5,0),
      liefermenge    number(5,0),
      gesamtpreis    number(10,2),
      );
```

Beispielhafte Umsetzung Teil3

```
-- Anlegen eines Typs als eingebettete
-- Tabelle von Bestellpositionen

CREATE OR REPLACE TYPE ntyp_position
         AS TABLE OF otyp_position;
/

CREATE OR REPLACE TYPE otyp_bestellung AS OBJECT (bestell_nr
number(6,0),
        bestellkunde REF otyp_kunde,
        bestelldatum date,
        lieferdatum  date,
        positionen   ntyp_position,
        rechnungsbetrag number(10,2)
        );
```

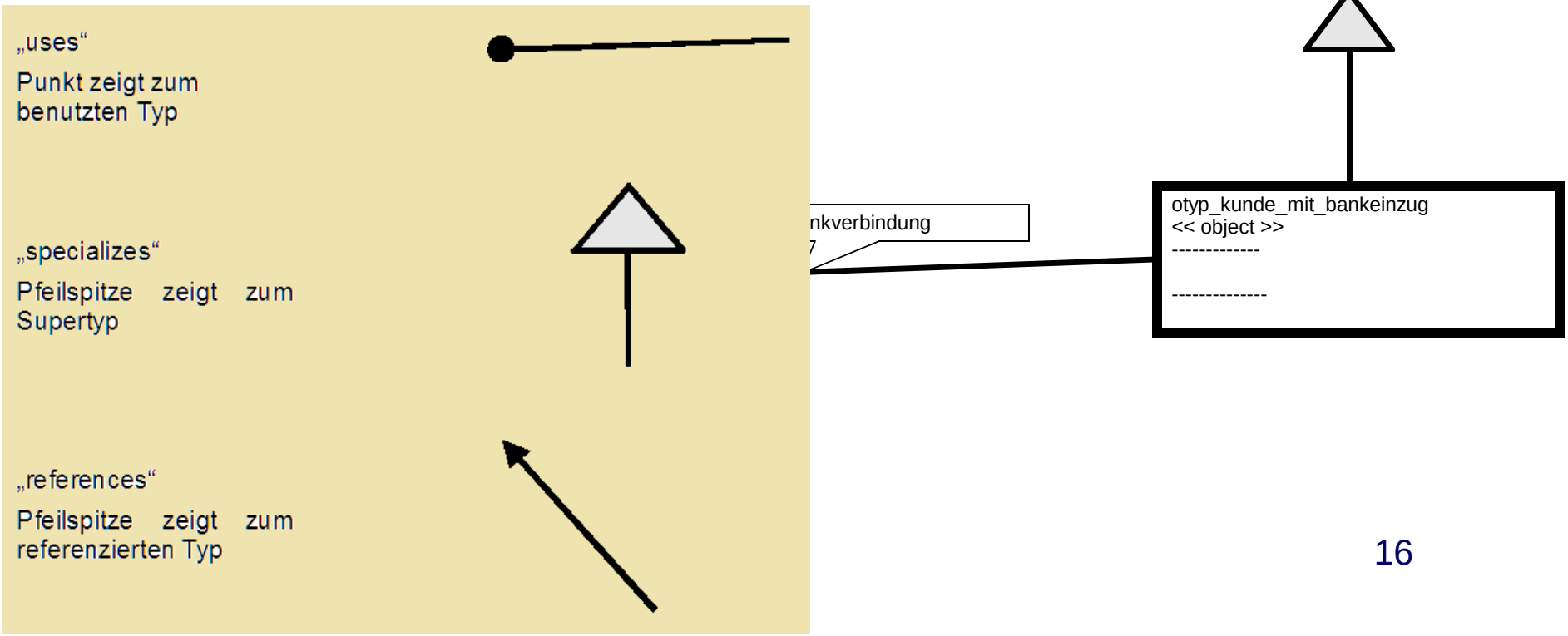
Plädoyer für ein eigenes Datentypmodell

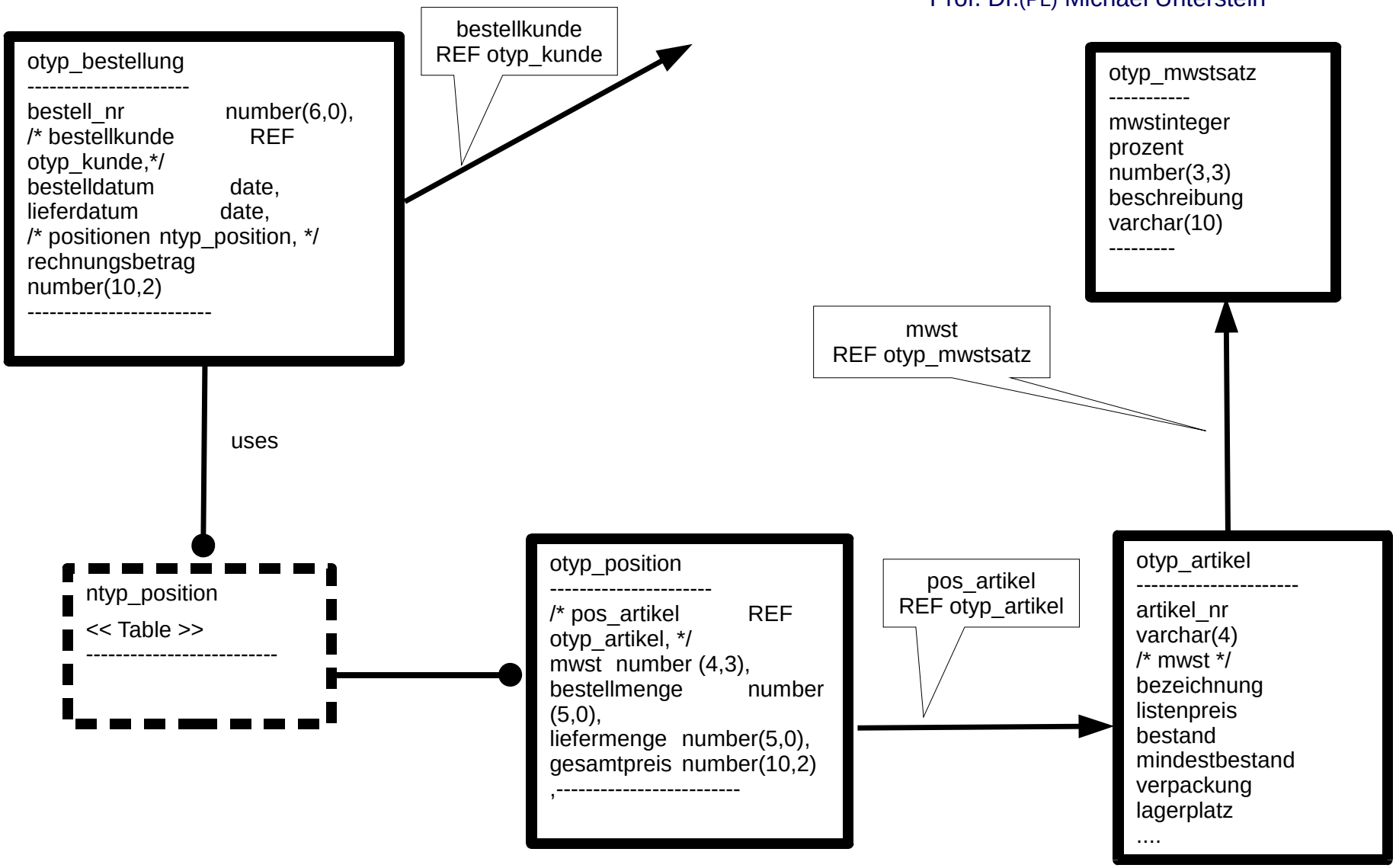
- Auf Basis eines abstrakten Datentyps können mehrere verschiedene Tabellen als Extent erzeugt werden.
- zwischen Typen sind andere (weniger) Beziehungen möglich als zwischen Klassen (im Wesentlichen „verwendet“, referenziert, „spezialisiert“)
- Die Beziehungen zwischen Extents sind mengenmäßig zu verstehen, Beziehungen zwischen Typen nicht. *Die Semantik von Typmodell und Extentmodell sind unterschiedlich.*
- unter SQL sind CREATE TYPE und CREATE TABLE ... OF zwei verschiedene Operationen, die in dieser Reihenfolge auszuführen sind.

Plädoyer für ein eigenes Datentypmodell 2

- Jedes Modell für sich wird übersichtlicher, als wenn im Klassenmodell ein Attribut wie „persistent“ benutzt (vgl. IDE Power Designer)
- Wichtige Eigenschaften des Datenmodells wie Primary Keys und weitere deklarative Einschränkungen können nur auf Basis des Extents festgelegt werden. (Mangel an SQL!).
- grafische Visualisierung der Abhängigkeiten von Typen zwecks Überblick über die Datenstrukturen Das Klassendiagramm leistet dies nicht.

Vorschlag Datentypendiagramm 1




otyp_bestellung
bestell_nr number(6,0),
/* bestellkunde REF otyp_kunde,*/
bestelldatum date,
lieferdatum date,
/* positionen ntyp_position, */
rechnungsbetrag number(10,2)
bestellkunde REF otyp_kunde
uses
ntyp_position
<< Table >>
otyp_position
/* pos_artikel REF otyp_artikel, */
mwst number (4,3),
bestellmenge number (5,0),
liefermenge number(5,0),
gesamtpreis number(10,2)
pos_artikel REF otyp_artikel
otyp_artikel
artikel_nr varchar(4)
/* mwst */
bezeichnung
listenpreis
bestand
mindestbestand
verpackung
lagerplatz
....
mwst REF otyp_mwstsatz
otyp_mwstsatz
mwstinteger prozent number(3,3)
beschreibung varchar(10)

Wer schon eine objektrelationale Datenbank hat, braucht eine Dokumentation derselben

- welche Typen gibt es, wie sind sie strukturiert und wie werden sie benutzt
- Ziel ist eine geschlossene Darstellung aller Abhängigkeiten der selbstdefinierten Typen
- welche Typen werden von welchen anderen Typen
  - benutzt
  - referenziert
  - spezialisiert

So soll es aussehen ...

```
TYPENAME                                  REFERENCED_NAME       DEPE  LEVEL
----------------------------------------- --------------------- ---- ------
OTYP_ADRESSE                                                          1
      OTYP_KUNDE                          OTYP_ADRESSE          HARD     2
            OTYP_BESTELLUNG               OTYP_KUNDE            REF      3
            OTYP_KUNDE_MIT_BANKEINZUG     OTYP_KUNDE            SPEC     3
      OTYP_KUNDE_MIT_BANKEINZUG           OTYP_ADRESSE          HARD     2
OTYP_GIROKONTO                                                        1
      OTYP_KUNDE_MIT_BANKEINZUG           OTYP_GIROKONTO        HARD     2
OTYP_MWSTSATZ                                                         1
      OTYP_ARTIKEL                        OTYP_MWSTSATZ         REF      2
            OTYP_POSITION                 OTYP_ARTIKEL          REF      3
                  NTYP_POSITION           OTYP_POSITION         HARD     4
                        OTYP_BESTELLUNG   NTYP_POSITION         HARD     5
VARR_TELEFONLISTE                                                     1
      OTYP_KUNDE                          VARR_TELEFONLISTE     HARD     2
            OTYP_BESTELLUNG               OTYP_KUNDE            REF      3
            OTYP_KUNDE_MIT_BANKEINZUG     OTYP_KUNDE            SPEC     3
      OTYP_KUNDE_MIT_BANKEINZUG           VARR_TELEFONLISTE     HARD     2
```

... wie kriegt man es hin ?

Benutzte Systemtabellen

TYPE_NAME	TYPECODE	SUPERTYPE_NAME
OTYP_KUNDE_MIT_BANKEINZUG	OBJECT	OTYP_KUNDE
OTYP_BESTELLUNG	OBJECT	(null)
NTYP_POSITION	COLLECTION	(null)
OTYP_POSITION		
OTYP_ARTIKEL		
OTYP_MWSTSATZ		
OTYP_GIROKONT		

TYPE_NAME	ATTR_NAME	ATTR_TYPE_NAME
OTYP_KUNDE	KUNDEN_NR	NUMBER
OTYP_KUNDE	STATUS	VARCHAR2

NAME	TYPE	REFERENCED_NAME	DEPENDENCY_TYPE
OTYP_POSITION	TYPE BODY	STANDARD	HARD
OTYP_POSITION	TYPE	STANDARD	HARD
OTYP_POSITION	TYPE	OTYP_ARTIKEL	REF
OTYP_POSITION	TYPE BODY	OTAB_ARTIKEL	HARD
OTYP_POSITION	TYPE BODY	OTYP_POSITION	HARD
UPDATE_ORDER_TOTAL	TRIGGER	DEMO_ORDER_ITEMS	HARD
UPDATE_ORDER_TOTAL	TRIGGER	DEMO_ORDERS	HARD
USER_DEPENDENCY_EXT	VIEW	USER_TYPES	HARD
USER_DEPENDENCY_EXT	VIEW	USER_DEPENDENCIES	HARD

USER_DEPENDENCIES

Ein View, der nur die interessierenden Daten enthält USER_DEPENDENCY_EXT2

NAME	REFERENCED_NAME	DEPENDENCY_TYPE
NTYP_POSITION	OTYP_POSITION	HARD
OTYP_ADRESSE	(null)	(null)
OTYP_ARTIKEL	OTYP_MWSTSATZ	REF
OTYP_BESTELLUNG	NTYP_POSITION	HARD
OTYP_BESTELLUNG	OTYP_KUNDE	REF
OTYP_GIROKONTO	(null)	(null)
OTYP_KUNDE	OTYP_ADRESSE	HARD
OTYP_KUNDE	VARR_TELEFONLISTE	HARD
OTYP_KUNDE_MIT_BANKEINZUG	OTYP_ADRESSE	HARD
OTYP_KUNDE_MIT_BANKEINZUG	OTYP_GIROKONTO	HARD
OTYP_KUNDE_MIT_BANKEINZUG	OTYP_KUNDE	SPEC
OTYP_KUNDE_MIT_BANKEINZUG	VARR_TELEFONLISTE	HARD
OTYP_MWSTSATZ	(null)	(null)
OTYP_POSITION	OTYP_ARTIKEL	REF
VARR_TELEFONLISTE	(null)	(null)

```
CREATE OR REPLACE VIEW user_dependency_ext2 AS
SELECT name, referenced_name,
      dependency_type
FROM USER_DEPENDENCIES ud1
WHERE TYPE = 'TYPE'
     AND referenced_owner = 'OOCHEF'
     AND type NOT LIKE '%BODY%'
     AND NOT EXISTS
        (SELECT * FROM user_types u2
         WHERE  ud1.name = u2.type_name
         AND    ud1.referenced_name =
                u2.supertype_name)
UNION
SELECT type_name, supertype_name, 'SPEC'
FROM   user_types
WHERE  supertype_name IS NOT NULL
UNION
SELECT type_name, NULL, NULL
FROM   user_types ut
WHERE  NOT EXISTS
          (SELECT *
           FROM USER_DEPENDENCIES ud
           WHERE ut.type_name = ud.name
           AND   type NOT LIKE '%BODY%'
           AND   referenced_owner = 'OOCHEF'
);
```

Diese Abfrage liefert das Ergebnis

```
SELECT LPAD(' ', 6*(level-1)) || name AS
  type_name, referenced_name,
  dependency_type, level
FROM user_dependency_ext2
START WITH referenced_name IS NULL
CONNECT BY PRIOR name = referenced_name
```

Literaturhinweis und Werbung