

Datendefinition in Oracle

Tabellendefinition: analog Standard → CREATE TABLE [eigentümer.]tabelle ... ;

→ die Tabelle wird im Schema des Eigentümers erstellt,
wenn Eigentümer = aktueller Nutzer, kann der Eigentümername entfallen

→ Aufruf des Tabelleninhaltes eines anderen Eigentümers:
SELECT * FROM eigentümer.tabelle

Abruf der Tabellenstruktur: DESC[RIBE] tabellenname;

Arbeit mit Synonymen → einfacher Zugriff auf Objekte

- Verkürzung langer Namen
- Verweis auf eine Tabelle eines Eigentümers durch einen anderen Nutzer

CREATE SYNONYM synonymname FOR object;

Definition und Änderung weiterer Datenbankobjekte über CREATE OR REPLACE
(vor einer Objektänderung ist kein DROP notwendig)

z.B. CREATE OR REPLACE VIEW viewname AS ...;

Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

Folie 3.1 

Oracle SQL Datentypen


VARCHAR2 (n)	Variable Zeichenkette der maximalen Länge n, n zwischen 1 und 4000
VARCHAR (n)	wie VARCHAR2
CHAR (n)	Feste Zeichenkette von n Byte, n zwischen 1 und 2000
NCHAR, NVARCHAR	Zeichenketten mit anderem Zeichensatz als dem der Datenbank
NUMBER (p, s)	p von 1 bis 38 (Gesamtzahl der Stellen) und s von -84 bis 127 (Vor- bzw. Nachkommastellen)
DATE	Gültiger Datumsbereich von -4712 bis 31.12.9999 enthält immer auch die sekundengenaue Uhrzeit
LONG	Variable Zeichenkette bis zu 2 GB
RAW (n)	Binärdaten der Länge n, n zwischen 1 und 2000 Bytes
LONG RAW	Binärdaten bis zu 2 GB
CLOB	Zeichenketten bis 4 GB
BLOB	Binärdaten bis 4 GB
CFILE, BFILE	Zeiger auf Dateien (Text, Binär)

Alle **ANSI-Datentypen** sind verfügbar und werden auf die obigen Datentypen abgebildet
(z.B. **CHARACTER, DECIMAL, INTEGER, FLOAT**).

Die Datentypen für unstrukturierte Daten (LONG, LONG RAW, LOBs) unterliegen starken
Einschränkungen. Die Manipulation solcher Objekte ist nur mit einer Prozedur und nicht
mit Transact-SQL möglich (PL/SQL, OCI).

Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

Folie 3.2 

Arbeit mit Systemtabellen des Data Dictionary

Anzeige von **Tabellen** eines Nutzers:

```
SELECT * FROM sys.tab;  
SELECT * FROM user_tables;
```

Anzeige der **Feldeigenschaften** einer Tabelle (diese in Großbuchstaben !!!):

```
SELECT * FROM COL WHERE TNAME = 'KUNDE';
```

Anzeige von **Integritätsbedingungen** einer Tabelle bzw. eines Eigentümers:

```
SELECT * FROM user_constraints where table_name = 'KUNDE';  
SELECT * FROM user_constraints where owner = 'MEIER';
```

Anzeige aller **Indizes** einer Tabelle:

```
SELECT * FROM user_indexes WHERE table_name = 'Kunde';
```

Anzeige aller an den Indizes einer Tabelle beteiligten Spalten:

```
SELECT * FROM user_ind_columns WHERE table_name = 'Kunde';
```

Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

Folie 3.3



Sequenz

➤ Automatisch generierte Abfolge von Zahlen

➤ Typische Anwendung: Erzeugung eines Primärschlüssels (Autoincrement)

Syntax:

```
CREATE SEQUENCE sequence  
INCREMENT BY n  
[START WITH n]  
[{MAXVALUE n | NOMAXVALUE}]  
[{MINVALUE n | NOMINVALUE}] ...;
```

„Pseudospalten“: NEXTVAL nächster möglicher Sequenzwert
CURRVAL gerade verwendeter Sequenzwert

Beispiel: Definition: CREATE SEQUENCE sBestellNr
INCREMENT BY 1
START WITH 1000;

Verwendung für automatische Vergabe von Bestellnummern:

```
INSERT INTO Bestell (BestellNr, Kunr, Datum)  
VALUES (sBestell.NEXTVAL, 123, to_date( '22.11.2010'));
```

Anzeige des verwendeten Sequenz-Wertes:

```
SELECT sBestellNr.CURRVAL FROM dual;
```

Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

Folie 3.4



Spezifika bei Abfragen

Kombination von Zeichenkettenspalten

```
SELECT Kunr, RTRIM(Vorname) || ' ' || Name Kundenname
FROM Kunde
```

RTRIM – Entfernen aller nach einer Eintragung stehenden Leerzeichen

Unterabfragen in der Spaltenliste

```
SELECT Artnr, Bezeichnung,
       (SELECT SUM(Menge*Vpreis)
        FROM Kauf k
        WHERE k.Artnr = a.Artnr) Umsatz      → Umsatz = Überschrift
FROM Artikel a;
```

Abgeleitete Tabellen

```
SELECT Name, AVG(Menge*Vpreis) "Druchschn.Umsatz je Kunde"
FROM (SELECT Name, Menge, Vpreis, Verkauf.Kunr
      FROM Kunde INNER JOIN Verkauf
      ON Kunde.Kunr=Verkauf.Kunr)
GROUP BY Verkauf.Kunr;
```

Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

Folie 3.5



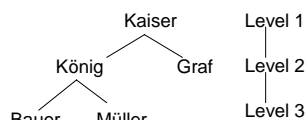
Hierarchische Abfragen

- Abbildung von Baumstrukturen in beliebiger Tiefe
- Abfragen auf rekursive Objekte

Syntax:

```
SELECT <attributliste>
FROM tabelle
CONNECT BY PRIOR <übergeordnetes Attribut> = <untergeordn. Attribut>
START WITH <attribut> = <startwert>;
```

Beispiel für hierarchische Abfragen:



Mitnr	Name	Chef
111	Kaiser	
210	König	111
220	Graf	111
301	Bauer	210
302	Müller	210

Ausgabe der Hierarchie mit Anzeige der Ebene

```
SELECT LEVEL, LPAD(' ',3*LEVEL-3) || Name, Mitnr
FROM Mitorg
CONNECT BY PRIOR Mitnr=Chef
START WITH Chef IS NULL;
```

Ergebnis:

LEVEL	Name	Mitnr
1	Kaiser	111
2	König	210
3	Bauer	301
3	Müller	302
2	Graf	220

Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

Folie 3.6



Ausgewählte SQL-Funktionen in Oracle

Konvertierungsfunktionen

to_char(a[,fmt]) Umwandlung der Zahl in eine Zeichenkette je nach Format fmt.
to_char(d[,fmt]) Umwandlung des Datums d in eine Zeichenkette je nach Format fmt.
to_date(s[,fmt]) Umwandlung der Zeichenkette s in ein Datum
to_number(s[,fmt]) Umwandlung der Zeichenkette s in eine Zahl

Ausgewählte Formatzeichen (fmt)

DD	Tag des Monats (1 - 31)	9	Zahl 0 bis 9 ohne führende Null
DAY	Name des Tages ('MONTAG' bis 'SONNTAG')	0	Zahl 0 bis 9 mit führender Null
Day	Name des Tages ('Montag' bis 'Sonntag')	S	Vorzeichen + oder -
MM	Monat des Jahres (1 - 12)	D	Dezimalpunkt
Mon	Monatsname dreistellig ('Jan' bis 'Dez')	G	Tausenderpunkt
Month	Monatsname ('Januar' bis 'Dezember')		
YY	Jahr zweistellig (00 bis 99)		
YYYY	Jahr vierstellig		
HH24	Uhrzeit: Stunde (0 - 24)		
MI	Uhrzeit: Minute (0-60)		
SS	Uhrzeit: Sekunde (0-60)		

Beispiele: Ermittlung der Verkäufe eines bestimmten Tages und der Umsätze je Kunde

```
SELECT * FROM Kauf
WHERE TO_CHAR (Kdatum, 'dd.mm.yyyy') = '24.12.2007';
```

```
SELECT TO_CHAR (Menge*VPreis, '999G990D00')
FROM Kauf GROUP BY Kunr
```

Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

Folie 3.7 

PL/SQL – Statements in Prozeduren und Triggern

Anweisungen für die Behandlung als Block gruppieren

```
BEGIN
  <anweisungsfolge>
END
```

Bedingte Ausführung von Befehlen

```
IF <bedingung>
  THEN <anweisungen für true>
[ELSIF <bedingung>
  THEN <anweisungen für true>
ELSE <anweisungen für false>]
END IF;
```

Einfache Schleife

```
LOOP
  <anweisungen>;
EXIT WHEN <bedingung>;
END LOOP;
```

FOR-Schleife (analoge Syntax WHILE)

```
BEGIN FOR <bedingung>
  LOOP
    <anweisungen>;
  END LOOP;
```

Weitere Anweisungen:

Variablenzuweisung: <variable> := <ausdruck>;

bzw. DECLARE <variable> := <ausdruck>;

Ausgabe von Texten/Meldungen: DBMS_OUTPUT.PUTLINE('text:' || variable);

Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

Folie 3.8 

Stored Procedures

Erstellen einer Stored Procedure:

```
CREATE [OR REPLACE] PROCEDURE proc_name
  [argument mode datentyp]
IS
  anweisungs_gruppe;
/
```

Beispiel einer Stored Procedure:

```
CREATE OR REPLACE PROCEDURE proc1
  (v_kunr IN Kunde.Kunr%TYPE,
   v_name OUT Kunde.Name%TYPE,
   v_vorname OUT Kunde.Vorname%TYPE)
IS
BEGIN
  SELECT Name, Vorname
  INTO v_name, v_vorname
  FROM Kunde
  WHERE Kunr = v_kunr;
END proc1;
/
```

Blockstruktur von PL/SQL

Deklarationsteil
DECLARE

Ausführungsteil
BEGIN ... END

Ausnahmeabschnitt
EXCEPTION
WHEN Fehler THEN Aktion

Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

Folie 3.9 

Cursor in PL/SQL

Syntax:

```
DECLARE CURSOR <Cursorname> IS
  <Select-Komponente>;

OPEN <Cursorname>;

FETCH <Cursorname> INTO <Variablenliste>;

CLOSE <Cursorname>;

END;
```

Beispiel für Cursornutzung:

```
DECLARE CURSOR c1 IS
  SELECT Name, Ort FROM Kunde;
  v_name Kunde.Name%TYPE;
  v_ort Kunde.Ort%TYPE;
OPEN c1;
LOOP
  FETCH c1 INTO v_name, v_ort;
  EXIT WHEN c1%NOTFOUND;
  ... (Datenmanipulation ...)
END LOOP;
CLOSE c1;
END;
```

← Variablen übernehmen Datentyp aus Tabellendef.

← %NOTFOUND ist true, wenn keine Tupel in c1 sind

Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

Folie 3.10 

Trigger in PL/SQL

Syntax: CREATE [OR REPLACE] TRIGGER <trigger-name>
 BEFORE | AFTER
 {INSERT | DELETE | UPDATE} [OF <column-list>]
 [OR {INSERT | DELETE | UPDATE} [OF <column-list>]]
 ...
 [OR {INSERT | DELETE | UPDATE} [OF <column-list>]]
 ON <table>
 [REFERENCING OLD AS <name> NEW AS <name>]
 [FOR EACH ROW]
 [WHEN (<condition>)]
 <pl/sql-block>;

* BEFORE, AFTER: Trigger wird vor/nach der auslösenden Operation ausgeführt.

* OF <column> (nur für UPDATE) schränkt Aktivierung auf angegebene Spalte ein.


* Zugriff auf Zeileninhalte vor und nach der Ausführung der aktivierenden Aktion
 mittels :OLD bzw. :NEW. (Aliasing durch REFERENCING OLD AS ... NEW AS ...).
 Schreiben in :NEW-Werte nur mit BEFORE-Trigger.

* FOR EACH ROW: Row-Trigger, sonst Statement-Trigger.

* WHEN (<condition>): zusätzliche Bedingung; OLD und NEW sind in <condition> erlaubt.

Prof. Dr. oec. G. Gräfe
 Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
 Logische Datenmodelle

Folie 3.11 

Beispiel eines Triggers (1)

Ausführungssequenz eines Triggers

Table Kauf

Kunr	Artnr	KDatum
123	1234	13.01.08
234	1234	15.01.08

← Before Statement Trigger

← Before Row Trigger

← After Row Trigger

← After Statement Trigger


Beispiel für Statement Trigger:

Test, ob KDatum auf einem Samstag oder Sonntag liegt, dann Meldung

```
CREATE OR REPLACE TRIGGER Datumtest BEFORE INSERT ON Kauf
BEGIN
    IF (TO_CHAR (Kdatum,'DAY') IN ('SAMSTAG','SONNTAG'))
    THEN
        RAISE_APPLICATION_ERROR (-20500, 'Verkauf erfolgt nicht am Wochenende');
    END IF;
END;
/
```

Prof. Dr. oec. G. Gräfe
 Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
 Logische Datenmodelle

Folie 3.12 

Beispiel eines Triggers (2)

Beispiel für AFTER-Row Trigger:

Nach jeder Preisänderung in der Tabelle Artikel wird in einer Tabelle Artikelprot die Artikelnummer, der alte und neue Preis sowie das aktuelle Datum hinterlegt

```
CREATE OR REPLACE TRIGGER Preisaend
  AFTER UPDATE OF EPreis ON Artikel
  FOR EACH ROW
  BEGIN
    INSERT INTO Artikelprot
      VALUES(:NEW.Artnr, :OLD.EPreis, :NEW.EPreis, SYSDATE);
  END;
```

Beispiel für BEFORE-Row Trigger mit EXCEPTION:

Bei der Eingabe einer neuen Bestellung in eine Tabelle Bestell (Bestellnr, Kunr, Datum) wird die Kundennummer auf referentielle Integrität getestet (dieser Test auch bei Änderung) sowie die Bestellnummer automatisch vergeben

Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

Folie 3.13 

Beispiel eines Triggers (3)

Beispiel für BEFORE-Row Trigger mit EXCEPTION:

```
CREATE OR REPLACE TRIGGER Bestell_eingabe
  BEFORE INSERT OR UPDATE OF Kunr ON Bestell
  FOR EACH ROW
  DECLARE
    v_falsche_Nr EXCEPTION;
    v_zahl INT;
  BEGIN
    IF INSERTING THEN
      SELECT COUNT(*) INTO v_zahl FROM Kunde k WHERE k.Kunr = :NEW.Kunr;
      IF v_zahl = 0 THEN
        RAISE v_falsche_Nr;
      END IF;
      SELECT sBestellnr.nextval INTO :NEW.Bestellnr FROM dual;
    END IF;
    IF UPDATING THEN
      SELECT COUNT(*) INTO v_zahl FROM Kunde k WHERE k.Kunr = :NEW.Kunr;
      IF v_zahl = 0 THEN
        RAISE v_falsche_Nr;
      END IF;
    END IF;
  EXCEPTION
    WHEN v_falsche_Nr THEN
      raise_application_error (-20000, 'Kundennummer nicht vorhanden');
  END;
```

Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

Folie 3.14 

Datenmodellierung

Die Datenmodellierung ist ein zweistufiger Abstraktionsprozess:

1. Ableitung des konzeptionellen Schemas aus dem Diskursbereich (*Fachkonzept der Datenbank*); Darstellung in einer der Fachsprache des Diskursbereiches nahen (semi-) formalen Notation (semantisches Datenmodell); unabhängig vom anzuwendenden Datenbankverwaltungssystem



2. Entwurf einer logischen Datenstruktur (logisches Datenmodell), die das konzeptionelle Schema widerspiegelt, und die physisch umgesetzt werden kann (*DV-Konzept der Datenbank*); abhängig vom anzuwendenden Datenbankverwaltungssystem

*semantisches
Datenmodell*



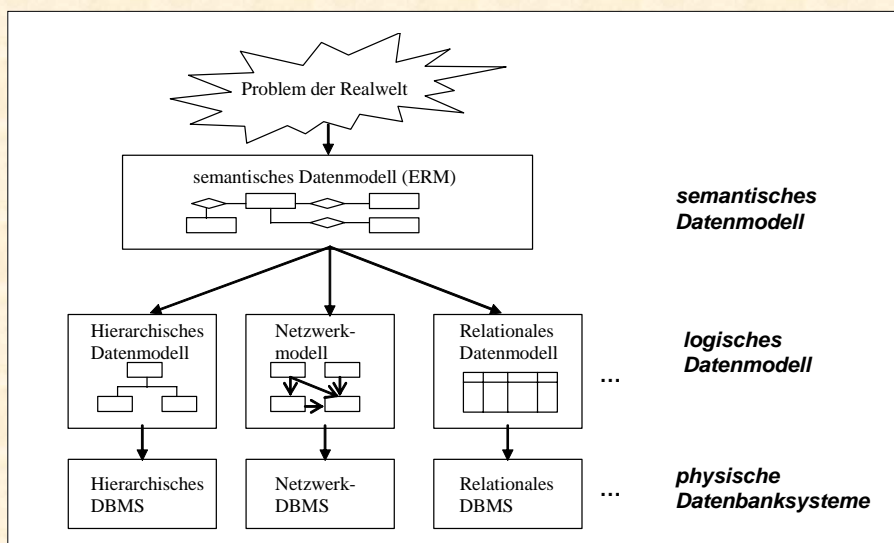
*logisches
Datenmodell*

Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

Folie 3.15

Zusammenhang zwischen semantischen und logischen Datenmodellen



Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

Folie 3.16

Semantische Datenmodelle

Die Aufgabe von **semantischen Datenmodellen** besteht darin, einen Ausschnitt aus der „Realität“ zu beschreiben. Dabei wird ein eher statisches Modell beschrieben. Durch die Darstellung der Objekte und Beziehungen werden die Zusammenhänge und Bedeutungen, sowie die Informations-Arten der einzelnen Objekte dargestellt.

Diese Art von Datenmodellen wird auch als infologisches Datenmodell bezeichnet. Der am häufigsten angewandte Model ist das ERM.

(siehe Modul Datenbanksysteme I)

Wesentliche Begriffe im ERM (Wdh.)

- Entity Objekt der realen Welt
- Entity-Typ Zusammenfassung von Objekten mit strukturellen Ähnlichkeiten
- Attribut Merkmal zur Beschreibung eines Objektes
- Wert jedes Attribut kann Werte aus einem bestimmten Wertebereich annehmen
- Relationship Beziehung zwischen den Entity-Typen bzw. (Set) Entitys
- Typ (Set-Typ) Art des Relationship (Kardinalität)

Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

Folie 3.17 

logische Datenmodelle

Das **logische Datenmodell** dient vor allem zur Darstellung der Struktur der Informationen. Allerdings ist diese Struktur abhängig von der Art des Datenbanksystems.

Das logische Datenmodell ist der „nächste“ Schritt nach der Erstellung des semantischen Datenmodells. Hierbei werden, z.B. bei einer relationalen Datenbank, aus dem Objekten und Beziehungen des semantischen Modells, Tabellen mit Attributen und deren Verknüpfungen generiert.


(siehe Modul Datenbanksysteme I: 6 Regeln zur Überführung eines ERM in ein RDM).

Typische Arten logischer Datenmodelle

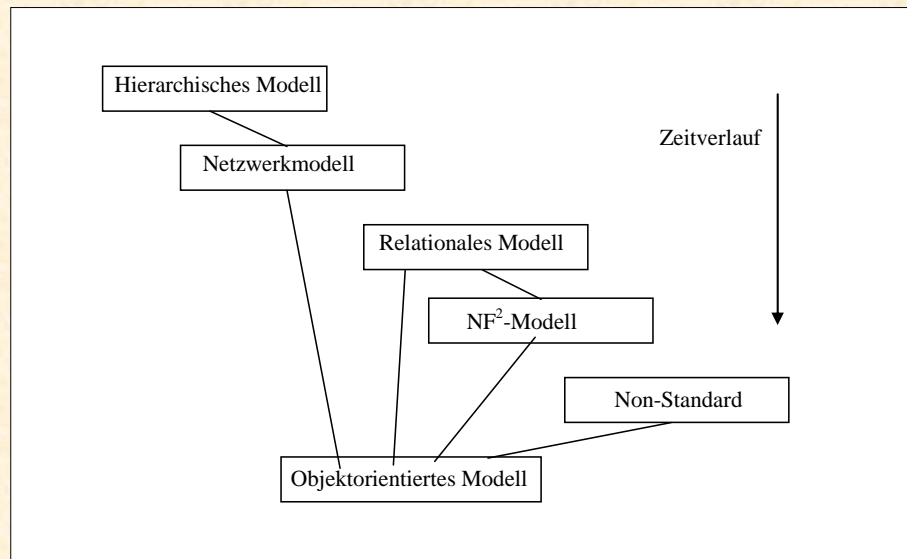
- Hierarchisches Datenmodell
 - Netzwerkdatenmodell
 - Relationales Datenmodell (RDM)
 - NF²-Modell
 - Objektorientiertes Datenmodell
 - "Objektrelationales Datenmodell"
- } *klassische logische Datenmodelle*

Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

Folie 3.18 

Logische Datenmodelle



Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

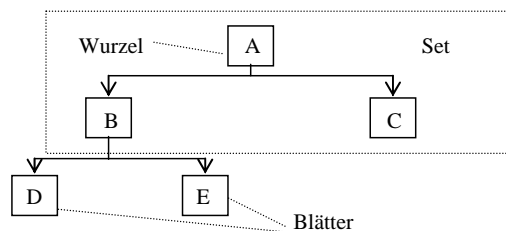
Folie 3.19

Hierarchisches Datenmodell (1)

Das hierarchische Datenmodell besitzt die folgenden Merkmale:

- Es gibt eine Menge von Entity-Typen und Beziehungen zwischen ihnen, die zusammen eine Ansammlung von Bäumen bilden.
- Jedes Entity gehört zu einem Entity-Typ, jedes Set zu einem Set-Typ. Dabei haben Set-Typen keine Attribute. Sets und Entities sind zeitlich veränderlich.
- Die Mitglieder eines Sets sind hierarchisch geordnet.
- In jedem Baum existiert genau ein Entity, das keinen Vorgänger hat. Dieses wird als Wurzel bezeichnet.
- Jedes Entity (außer Wurzel) hat genau einen Vorgänger (eindeutige Vorgänger-Nachfolger-Beziehung) und kann mehrere Nachfolger haben (außer Blätter).
- Jeder Zweig endet mit Blättern, die keine Nachfolger haben.
- Jedes Entity ist von der Wurzel ausgehend über einen gerichteten Graph erreichbar.

Prinzipiskizze:



Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

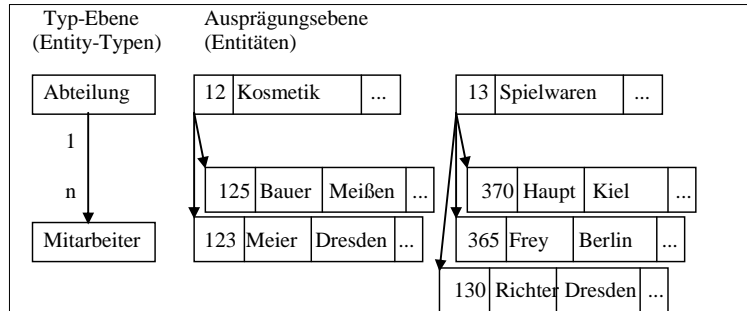
Datenbanksysteme II
Logische Datenmodelle

Folie 3.20

Hierarchisches Datenmodell (2)

Bei der grafischen Darstellung wird die Baumstruktur in einen gerichteten Graphen umgesetzt, wobei zwischen einem Graphen zur Darstellung der Typebene (Entity-Typen) und einem Graphen der Ausprägungsebene (Entities) zu unterscheiden ist.

Beispiel:



Der Wurzelknoten wird durch die jeweilige Abteilung gebildet. Die Abteilung *Kosmetik* ist PARENT und besitzt mehrere CHILDS: *Bauer* und *Meier*. Die Mitarbeiter bilden die Blätter und besitzen somit keine Nachfolger. Jede dieser Beziehungen bilden ein Set.

Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

Folie 3.21 

Netzwerkdatenmodell (1)

- Zeitlich zwischen hierarchischem und relationalem Modell entwickelt
- In der Microcomputerwelt wenig verbreitet.

Grundprinzip: Abbildung der Datenstruktur des ERM auf Netzstrukturen

Das Netzwerkdatenmodell besitzt die folgenden Merkmale:

- Es gibt eine Menge von Entity-Typen und Beziehungen zwischen ihnen, die zusammen einen beliebigen gerichteten Graphen bilden.
- Jedes Entity gehört zu einem Entity-Typ, jedes Set zu einem Set-Typ. Dabei haben Set-Typen keine Attribute. Sets und Entities sind zeitlich veränderlich.
- Die Mitglieder eines Sets sind geordnet.
- In jedem Baum existiert genau ein Entity, das keinen Vorgänger hat. Dieses wird als Wurzel bezeichnet.
- Jedes Entity kann sowohl mehrere als auch mehrere Nachfolger haben.
- Jeder Zweig endet mit Blättern, die keine Nachfolger haben.
- Jedes Entity ist von der Wurzel ausgehend über jeden, nicht notwendig gerichteten Weg erreichbar.

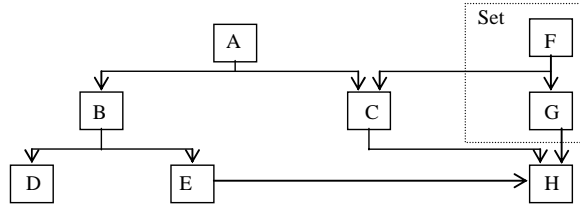
Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

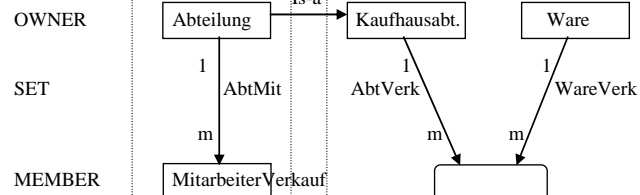
Folie 3.22 

Netzwerkdatenmodell (2)

Prinzipiskizze:



Modellierung von SET-Typen
im Netzwerkmodell:



Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

Folie 3.23

Relationenbegriff (Wdh.)

Mathematische Definition

Sind W_1, W_2, \dots, W_n nichtleere Mengen, dann ist jede nichtleere Teilmenge der Produktmenge

$$PM = W_1 * W_2 * \dots * W_n \text{ eine } n\text{-stellige}$$

Relation $R \subseteq W_1 \times W_2 \times \dots \times W_n$

W_1, W_2, \dots, W_n sind die Wertebereiche (Menge aller Werte = Domäne) der Attribute A_1, A_2, \dots, A_n von Entities.
 n ist der Grad (degree) der Relation.

Darstellung als Tabelle

Mitarbeiter	MITNR	VORNAME	NAME	ANSCHRIFT	ALTER	← Attributnamen
Zeilen/ Tupel	101	Peter	Silie	Dresden	5	Attributwerte
	102	Mario	Nette	Dresden	6	
	103	Klaus	Uhr	Radebeul	20	
	104	Otto	Graffie	Freital	17	
	105	Kurt	Isane	Friedersdorf	35	
	106	Paul	Aner	Merseburg	25	
	↑	Nichtschlüsselspalten			Domäne	
	Primär- schlüssel	Datenspalten				

Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

Folie 3.24

Charakteristika des Relationalen Datenmodells (Wdh.)

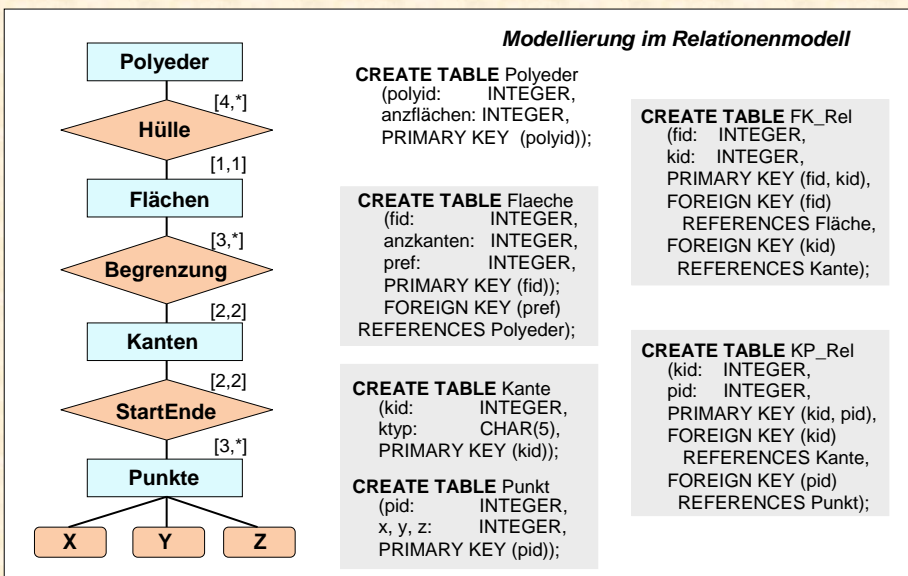
- K1: Es gibt eine Menge von Relationen unterschiedlichen Grades über den Attributwerten.
- K2: Die Relationen sind untereinander gleichberechtigt.
- K3: Die Relationen sind zeitlich veränderlich (Einfügen, Löschen, Ändern von Tupeln).
- K4: Jede Relation hat dabei charakteristische Eigenschaften.
- K41: Jedes Tupel der Relation kommt nur einmal vor.
- K42: Die Reihenfolge der Tupel ist beliebig.
- K43: Die Reihenfolge der Spalten ist auch beliebig, da die Bezugnahme auf die Spalten über die eindeutigen Attributnamen und nicht über eine Spaltennummer erfolgt.
Attributnamen müssen in einer Relation unterschiedlich sein.
Die Reihenfolge wird einmal vorgegeben, bleibt dann bestehen.
- K44: Es gibt genau einen Primärschlüssel, der die Tupel eindeutig identifiziert.

Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

Folie 3.25 

Aufwendige Modellierung von Polyedern im RM



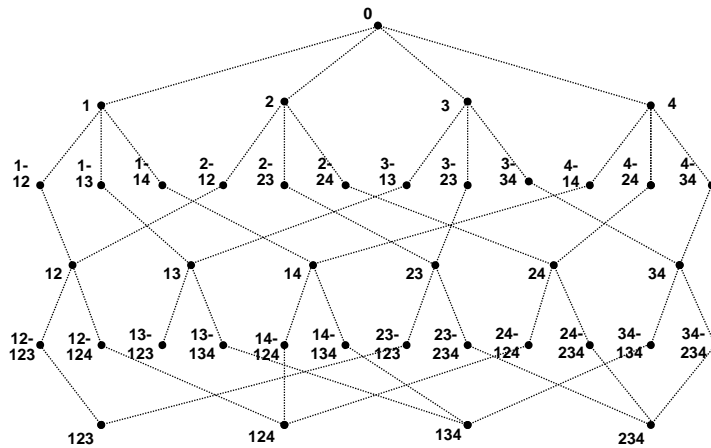
Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

Folie 3.26 

RM – angemessene Modellierung?

Darstellung eines Tetraeder mit vid = 0



Relationen

Polyeder

Fläche

FK-Rel

Kante

KP-Rel

Punkt

Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

Folie 3.27

Objektorientiertes Datenmodell

- zeitlich „jüngstes“ Datenmodell
- integriert Eigenschaften von klassischen logischen Datenmodellen mit denen objektorientierter Programmiersprachen

Grundprinzip: Objektorientierung

- ein Objekt in der Realität ist ein Objekt in der Datenbank
- Integration von Daten und Methoden (Funktionen);

- beruht auf (i. a. mehrstufiger) Klassifizierung der Objekte der Realwelt

Beispiel für eine mehrstufige Klassifizierung:

Klassifikationsbäume in der Botanik (Quelle: ROTHMALER, W.: Exkursionsflora)

Hierarchie	Beispiel	Attribut (A) /Merkmal (M)
Stamm	Samenpflanzen	A: Pflanzen mit echten Blüten M: Vermehrung durch Samen
Klasse	Nadelhölzer	A: Bäume oder Sträucher A: Blätter nadelartig oder schuppenförmig
Ordnung	Kiefernartige	A: Blüten in Zapfenform A: Samen nussartig, geflügelt
Familie	Zypressengewächse	A: Zapfen klein, holzig oder beerenartig
Art	Gemeiner Wacholder	A: Blätter alle nadelig, stechend M: Wuchs säulenförmig

Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

Folie 3.28

Charakterisierung der Objektorientierung (1)

1. Objektidentität

- Objekte (ERM: Entity)
 - beziehen sich auf Erscheinungen der Realwelt (Diskursbereich),
 - stellen eine abgegrenzte Einheit dar,
 - haben Eigenschaften (Datenaspekt),
 - haben Methoden, welche die Eigenschaften verändern können (Funktionsaspekt).
- Für jedes unabhängige, im Modell abgebildete Objekt muss eindeutige Identität garantiert werden (=> Objektidentifikator = ID)

2. Klassifizierung von Datenobjekten

- Gleichartige Objekte werden zu Klassen zusammengefasst (ERM: Entitytyp); Elemente von Klassen heißen „Instanzen“.
- Klassifizierung kann mehrstufig sein: Entstehung von Klassenhierarchien; Superklasse - Subklassen (ERM: Spezialisierung bzw. Generalisierung).

3. Kapselung von Operationen, Methoden und Persistenz

- Zusammenfassung von Attributen (Eigenschaften) und Methoden; Integration von Daten- und Funktionsaspekt.
- Attribute eines Objekts können nur durch Methoden des Objekts selbst, nicht „von außen“ verändert werden.

Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

Folie 3.29 

Charakterisierung der Objektorientierung (2)

4. Typhierarchien und Vererbung


- Jede Instanz erbt die Attribute und Methoden der Klasse (=> Ausprägungen).
- Jede Klasse erbt die Attribute und Methoden der übergeordneten Klasse.
- Eine Superklasse kann ihre Attribute und Methoden an mehrere Subklassen vererben, wobei jede Subklasse in der Regel zusätzliche Attribute und Methoden besitzt (einfache Vererbung; „Vererbungsbäume“).
- Eine Subklasse kann auch von mehreren Superklassen erben (multiple Vererbung; „Vererbungsnetze“).

5. Polymorphismus

- Ein Objekt kann von einem beliebigen Typ (= polymorph) sein und die Interpretation einer Methode hängt vom Typ ab.
- Es kann immer mehrere Objekte unterschiedlichen Typs geben, denen eine bestimmte Methode zugeordnet wird. Dabei muss der Datenbanknutzer nicht wissen, zu welchen Typ ein bestimmtes Objekt gehört.
=> statisches und dynamisches Binden

Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle


Folie 3.30 

Bedeutung des Objektorientierten Datenmodells

- gestattet Integration von Daten und Funktionen
- ist Grundlage für objektorientierte Systementwicklung
 - „Impedance Mismatch“ beim Zugriff auf relationale Datenbestände
 - Brüche im Typsystem
- Abbildung komplexer Objekte
 - rekursive Objekte => Stücklisten
 - 3D-Modelle
 - Spatiale Daten
 - nicht formatierte Daten (=> Multimedia)

Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

Folie 3.31 

Objektorientierten und Objektrelationales Datenmodell

Objektorientierten Datenmodell

- Implementierungen des objektorientierten Datenmodells
- erste Produkte: 1986/87
- Höhepunkt 1996: ca. 5 % des weltweiten Umsatzes, danach rückläufig
- Marktführer: Object Design mit ObjectStore
- BS-Umgebung: meist UNIX



Objektrelationales Datenmodell

- Erweiterung des relationalen Datenmodells um objektorientierte Konzepte

Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

Folie 3.32 

Objektrelationales Datenmodell

Erweiterung des relationalen Datenmodells um objektorientierte Konzepte bzw. Kombination beider Ansätze

- Beibehaltung der bewährten Konzepte relationaler DBS
- Datenbank weiterhin Menge von Relationen

- * eingeführt mit UniSQL (W. Kim)
- * popularisiert durch M. Stonebraker mit Illustra, dann an Informix verkauft
- * heute klare Entwicklungsrichtung fast aller relationalen DBMS (Oracle, DB2, Sybase, PostgreSQL)
- * Normung: ab SQL-3/SQL99-Standard

Ziele

- Anwendung von Datenbankfunktionalität auf "Nicht-Standard"-Daten
- Evolutionäre Erweiterung relationaler DBMS
 - * Integration objektorientierter Konzepte
 - * Aufwärtskompatibilität basierend auf SQL
- Vereinigung der Vorteile relationaler und objektorientierter DBMS

Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

Folie 3.33



NF² - Ausprägung (1 Tupel)

Volumen							
Vid	Bez	Flaechen	Kanten	Punkte			
		Fid	Kid	Pid	X	Y	Z
0	Tetraeder	1	12	123	0	0	0
				124	100	0	0
			13	123	0	0	0
				134	50	44	75
			14	124	100	0	0
				134	50	44	75
		2	12	123	0	0	0
				124	100	0	0
			23	123	0	0	0
				234	50	87	0
			24	124	100	0	0
				234	50	87	0
		3	13	123	0	0	0
				134	50	44	75
			23	123	0	0	0
				234	50	87	0
			34	134	50	44	75
				234	50	87	0
		4	14	124	100	0	0
				134	50	44	75
			24	124	100	0	0
				234	50	87	0
			34	134	50	44	75
				234	50	87	0

Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

Folie 3.34



Objektrelationale Features in Oracle

Oracle 9i-11g bietet folgende OO-Konzepte an:

- benutzerdefinierte Datentypen
- Methoden auf benutzerdefinierten Datentypen
- Objektidentität

Wesentliche (noch) fehlende OO-Konzepte:

- Vererbung
- Polymorphismus

→ Objektrelationale Features:

- * Abstrakte Datentypen
- * Verschachtelte Tabellen (Nested Tables)
- * Variable Arrays
- * Große Objekte
- * Referenzen
- * Objekt-Views


Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

Folie 3.35 

Komplexbeispiel für objektrelationale Feature

Tabelle Kunden

Attribut	Beispielwerte			
Kunr	123			
Name	Supermarkt123			
RAdresse Strasse PLZ Ort	Bergstrasse 12 01069 Dresden			
Beginn Dauer	12.11.2001 → SYSDATE - Beginn			
Lieferstellen	Nr.	Strasse	PLZ	Ort
	11 13	Waldweg 1 Austr. 34	01117 11012	Dresden Berlin
Einkaeufer	Claire Grube Anna Nass			
Lageplan				

Objekt,
Datentyp

Methode

Nested Table

Arrays

Large Object

Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

Folie 3.36 

Abstrakte Datentypen (1) - OBJECT

Datentypen, die aus einem oder mehreren Subtypen bestehen

```
CREATE [OR REPLACE] TYPE typename AS OBJECT  
( attribut1 datentyp1, attribut2 datentyp2 ...)
```

Merkmale:

- * Abstrakte Datentypen lassen sich beliebig verschachteln → Aufbau von Objekttabellen
- * Wiederverwendbarkeit
- * Einhaltung von Standards

Beispiel:

Datendefinition - Erzeugen/Verwenden abstrakter Datentypen:

```
CREATE TYPE AdresseTY AS OBJECT  
(Strasse CHAR(30),  
  PLZ CHAR(5),  
  Ort CHAR(20));
```

```
CREATE TABLE Kunde  
(Kunr INT,  
  Name CHAR(20),  
  Radresse AdresseTY);
```

Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

Folie 3.37 

Abstrakte Datentypen (2)

Datenmanipulationen in Attributen mit abstrakten Datentypen

→erfordern immer den Bezug zum Typnamen

```
INSERT INTO tabelle VALUES(..., typename(Wert), ...)
```

```
SELECT typename(attribut1, attribut2 ...) FROM tabelle
```

Beispiele:

1. Datenmanipulation -Eingabe von Datensätzen:

```
INSERT INTO Kunde  
(Kunr, Name, Radresse)  
VALUES  
(123, 'Meier', AdresseTY('Bergstr. 12', '01069', 'Dresden'));
```

2. Abfragen in abstr. Datentypen:

```
SELECT k.Radresse.PLZ, k.Radresse.Ort  
FROM Kunde k;
```

Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

Folie 3.38 

Objektviews (1)

(nachträgliches) Überlagern von bestehenden relationalen Tabellen mit objektorientierten Strukturen (z.B. abstr. Datentypen)

```
CREATE [OR REPLACE] VIEW viewname (spalte1, ...) AS
  SELECT {attribut1 bzw. datentyp1}, ...
  FROM tabelle
```

Beispiel:

Anlegen eines Views der Kundenorte:

```
CREATE VIEW KundenOV (Kundennr, Name, PLZ, Wohnort) AS
  SELECT Kunr, Name, k.Radresse.PLZ, k.Radresse.Ort
  FROM Kunde k;
```

Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

Folie 3.39 

Objektviews (2)

Manipulation in Views (z.B. Insert) erfordert die Vereinbarung eines Triggers

Beispiel:

Anlegen eines Triggers im View der Kundenorte:

```
CREATE OR REPLACE TRIGGER KundenOV_trig
  INSTEAD OF INSERT
  ON KundenOV
  FOR EACH ROW
  BEGIN
    IF INSERTING THEN
      INSERT INTO Kunde VALUES(:new.Kunr, :new.Name,
                                AdresseTY(:new.PLZ, :new.Ort));
    END IF;
  END;
```

Manipulation (Insert) erfolgt analog abstrakter Datentypen

Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

Folie 3.40 

Methoden (1)

Verknüpfung von Datentypen (DT) mit Methoden, die auf die Daten einer Tabelle angewendet werden

Erstellen eines abstrakten DT: CREATE [OR REPLACE] TYPE
Erstellen einer Methode: CREATE [OR REPLACE] TYPE BODY

Methoden werden im Datentyp als MEMBER FUNCTION hinterlegt

Beispiel: Methode zur Berechnung der Dauer der Kundenbeziehung aus dem Datum

1. Datentyp Dauer (DauerTY)

```
CREATE OR REPLACE TYPE DauerTY AS OBJECT  
(Beginn DATE,  
  MEMBER FUNCTION Dauer(Beginn DATE)  
  RETURN NUMBER);
```

2. Methode zur Berechnung

```
CREATE OR REPLACE TYPE BODY DauerTY AS  
  MEMBER FUNCTION Dauer(Beginn DATE)  
  RETURN NUMBER IS  
  BEGIN  
    RETURN ROUND(SYSDATE - Beginn);  
  END;  
END;
```

Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

Folie 3.41 

Methoden (2)

Beispiel - Anfügen der Methode an Tabelle Kunden:

```
ALTER TABLE Kunde ADD Beginn DauerTY;
```

Eingabe von Datensätzen:

```
INSERT INTO Kunde VALUES  
(125, 'Schmidt', AdresseTY('Seestr. 22', '01089', 'Dresden'),  
  DauerTY(TO_DATE('29.01.2009')));
```

Abfrage der Dauer der Kundenbeziehung:

```
SELECT K.Beginn.Dauer()  
FROM Kunde K;
```

Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

Folie 3.42 

Verschachtelte Tabellen (Nested Tables) – (1)

- * Tabelle in der Spalte einer anderen Tabelle
- * Für jeden Datensatz einer Haupttabelle wird in einer Spalte eine Sammlung von Zeilen angelegt
- * Abbildung einer 1:m-Beziehung

Beispiel: Tabelle Kunden

Kunr	Name	RAдресse	Lieferstelle
123	SuperM	11 Bergstr. 12
			12 Waldweg 3
			01069 Dresden
			01474 Radeburg

Syntax: CREATE TYPE subtabelle AS TABLE OF typename

Beispiel Tabelle Kunden - Erstellen Verkaufstelle:

```
CREATE TYPE LieferadresseTY AS OBJECT
  (Verknr INT,
   Strasse CHAR(30),
   PLZ CHAR(5),
   Ort CHAR(20));

CREATE TYPE LieferadresseNT AS TABLE OF LieferadresseTY;
```

Verschachtelte Tabellen (Nested Tables) – (2)

Anfügen einer Subtabelle in eine Tabellendefinition:

NESTED TABLE attributname STORE AS nt_name

nt_name = Name der Tabelle, in der NT gespeichert wird

Beispiel:

```
CREATE TABLE Kunde
  (Kunr INT,
   Name CHAR(20),
   RAдресse AdresseTY
   Lieferstelle LieferadresseNT
   NESTED TABLE Lieferstelle STORE AS NTLieferstellen);
```

oder

```
ALTER TABLE Kunde ADD
  Lieferstelle LieferadresseNT
  NESTED TABLE Lieferstelle STORE AS NTLieferstellen;
```

NTLieferstellen = Name der Tabelle, in der NT gespeichert wird

Verschachtelte Tabellen (Nested Tables) – (3)

Beispiele für Datenmanipulation in NT:

Einfügen von Datensätzen in eine NT (2 Schritte):

```
INSERT INTO Kunde (Kunr, Name, Lieferstelle)
VALUES (123, 'Meier', LieferadresseNT());
```

```
INSERT INTO TABLE (SELECT k.Lieferstelle FROM Kunde k WHERE k.Kunr='123')
VALUES (LieferadresseTY (11, 'Bergstr. 12', '01069', 'Dresden'));
```

Abfragen in NT sollten in PL/SQL (Stored Procedures/Trigger) unter Nutzung eines Cursors erfolgen.

→ Angabe der Spaltennamen im Format: Tabelle.Spalte.NT-Attribut

Heraussuchen aller Lieferstellen von Meier: (nur exemplarisch)

```
SELECT k.Name, l.*
FROM Kunde k, table(k.Lieferstelle) l
WHERE k.Name='Meier';
```

Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

Folie 3.45 

Variable Arrays (VARRAYS)

Analoge Funktionalität wie geschachtelte Tabellen

Unterschied: * alle Objekte eines VARRAY haben gleichen Datentyp
 * Eingeschränkte Größe (Zeilenzahl)

Problem: Abfrage in VARRAY nicht mit Select-Anweisung,
 nur über verschachtelte Cursor FOR-Schleifen in PL/SQL

Syntax: CREATE OR REPLACE TYPE arrayname
 AS VARRAY(anzahl) OF datentyp

Beispiele:

Anlegen eines Feldes Einkäufer: CREATE OR REPLACE TYPE EinkaufVA
AS VARRAY(5) OF VARCHAR2(25);

ALTER TABLE Kunde ADD Einkaeufer EinkaufVA;

Eingabe in Variable Arrays: INSERT INTO Kunde(Kunr, Name, Einkaeufer) VALUES
(130, 'Bauer GmbH',
EinkaufVA('Grube, Claire', 'Nass, Anna'));

Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

Folie 3.46 

Large Objects (LOBs) (1)

LOBs können einer von vier Datentypen sein:

- BLOBs**, die unstrukturierte binäre Daten speichern
- CLOBs**, die Zeichenfolgedaten speichern
- NCLOBs**, die Zeichenfolgedaten unter Verwendung eines landesspezifischen Zeichensatzes speichern (haupts. zur Unterstützung asiatischer Sprachen)
- BFILEs**, die binäre Dateien im Dateisystem referenzieren

Vorteile:

- * ein LOB kann 4 Gbyte oder die doppelte Kapazität der LONG- oder LONG RAW-Spalte aufnehmen.
- * eine Tabelle kann mehr als eine LOB-Spalte enthalten, jedoch nur eine LONG- oder LONG RAW-Spalte
- * die in einer LOB-Spalte gespeicherten Daten werden in einem anderen Bereich gespeichert als die Tabelle, welche die LOB-Spalte enthält, was zu einer besseren Gesamt-Performance führt.

Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

Folie 3.47 

Large Objects (LOBs) (2)

Datendefinition:

Beispiel Anlegen von LOB-Attributen in der Tabelle Kunde:

```
CREATE TABLE Kunde  
(Kunr INT,  
Name VARCHAR2(30),  
Anfahrtbeschreibung CLOB,  
Lageplan BFILE);
```

Anlegen eines Verzeichnis-ALIAS für BFILE:

```
CREATE DIRECTORY Stadtplaene AS 'C:\Plan';
```

Datenmanipulation:

Einfügen von Datensätzen:

```
INSERT INTO Kunde VALUES  
(223, 'Supermedia', EMPTY_BLOB(),  
BFILENAME('C:\Plan', 'Plan100.jpg'));
```

EMPTY-BLOB() - Leerer Locatorwert (Nullwert)

Nachtragen eines Textes für Anfahrtbeschreibung:

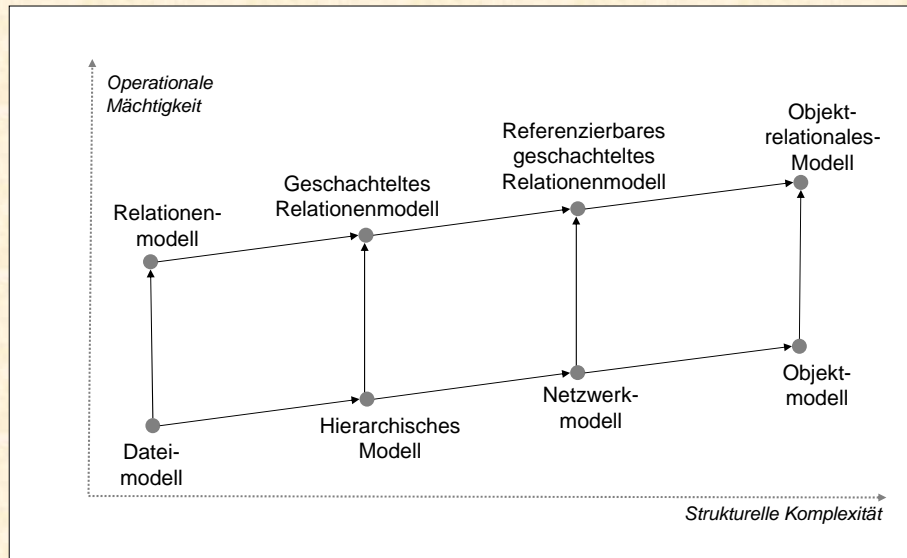
```
UPDATE Artikel  
SET Beschreibung='Anfahrt über A14, Ausfahrt Dresden- ...'  
WHERE Kunr=223;
```

Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

Folie 3.48 

Evolution von Datenmodellen

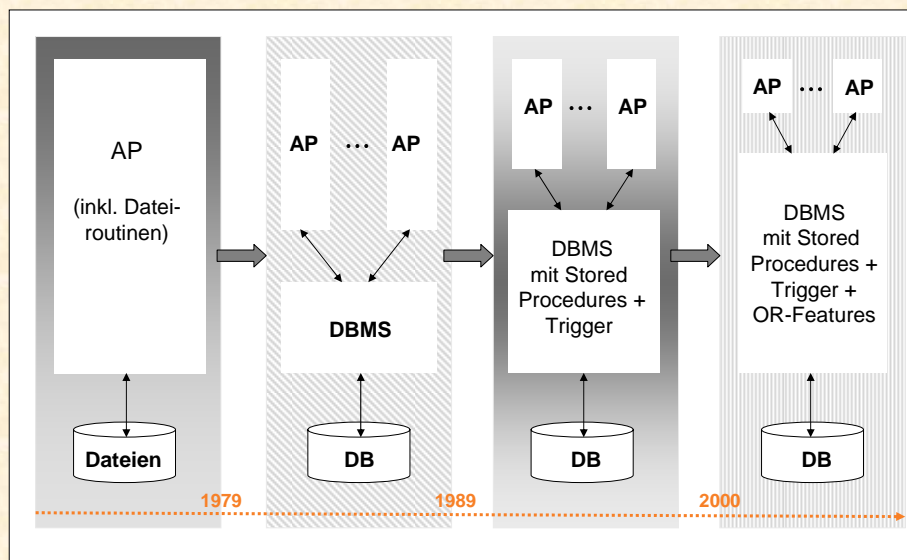


Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

Folie 3.49

Aufgabenverteilung im Wandel der Zeit



Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Logische Datenmodelle

Folie 3.50