9. Objektorientierung und Datenbanken

- **DBS-Markt**
- **Objektorientierung - Überblick**
- **Beschränkungen klassischer Datenmodelle**
  - Beispiel Landinformationssysteme
  - Beispiel CAD-Systeme
- **Grundkonzepte der Objektorientierung**
  - OODBS-Manifesto
  - Überblick über einige Konzepte
  - ODMG-Objektmodell: Überblick
  - Objektorientierte DBS
- **Objekt-relationale DBS - Vision**
  - verschiedene Systemarchitekturen
  - objekt-relationale DB-Technologie
  - Erweiterbarkeitsinfrastruktur

DBS-Markt

Datenstrukturen \ Anfragen	einfach	komplex
komplex	OO 10^8 $	OR ?
einfach	Dateisysteme Video-Server	RM (SQL) 10^{10} $

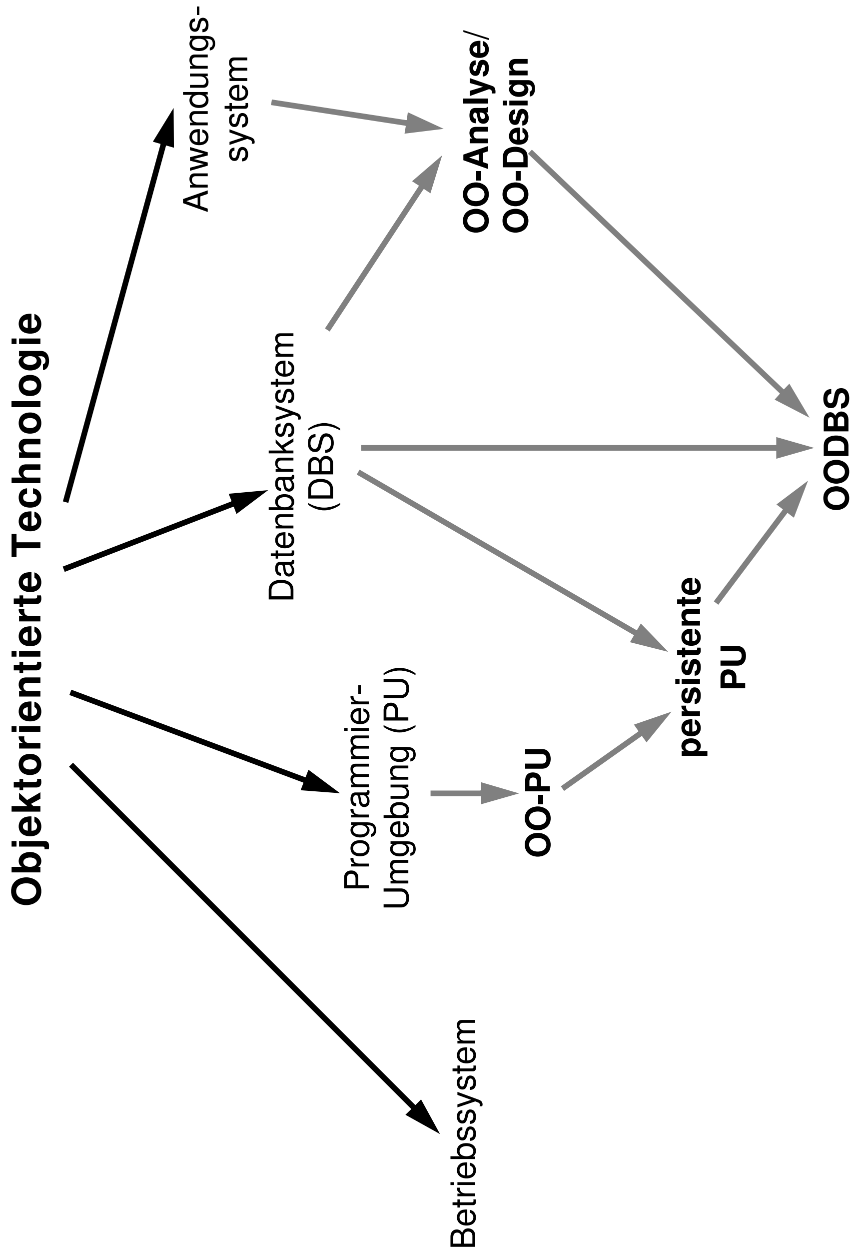
- **Einfache Daten, einfache Anfragen**
  - Datenstruktur ist dem System nicht bekannt
  - Künftig werden solche Systeme wahrscheinlich mit Anfrage-möglichkeiten (z. B. SQL) ausgestattet
- **Einfache Daten, komplexe Anfragen**
  - RDBS: skalierbar, robust, Zugriff über Struktur und Inhalt
  - Begrenzte Unterstützung für komplexe, als BLOBs gespeicherte Daten
  - RDBS können diese BLOBs nicht indexieren, manipulieren oder über ihren Inhalt suchen
- **Komplexe Daten, einfache Anfragen**
  - Persistente komplexe Objekte, die durch Java, C++, Smalltalk, ... manipuliert werden
  - Begrenzte Skalierbarkeit in Bezug auf große Datenvolumina und große Anzahlen von Benutzer
- **Komplexe Daten, komplexe Anfragen**
  - OR-Server können komplexe Daten als Objekte handhaben
  - Benutzerdefinierte Funktionen lassen sich zur Manipulation der Daten im Server heranziehen
  - Erweiterbarkeit ist für Datentypen und Funktionen möglich

Objektorientierung - Überblick

- **Dominierendes Thema**
  - Tagungen, Workshops und Tutorien
  - Zeitschriften und Büchern
  - Standardisierungsgremien
    - Object Management Group (OMG)
    - Object Database Management Group (ODMG)
    - SQL-Standardisierung (SQL4)
    - ...

- **Historie**
  - **Simula** (1967)
    (Klassenkonzept)
  - **Smalltalk** (1972)
    („Everything is an object“**,** Message Passing)
  - **zahlreiche weitere Programmiersprachen** (seit 1985**)**
    wie C++, EIFFEL, ...
  - **OODBMS** (seit 1986)
    - GemStone, IRIS, O2, Objectivity/DB, ObjectStore,
    - Orion (UniSQL), Poet, Versant, ...
  - **Objektorientierte Systementwicklung** (seit 1986)
    - GOOD, MOOD, HOOD, ...
  - **Einfluß auf Software-Technik**
    - Modularisierung
    - Geheimhaltungsprinzip (Information Hiding)
    - Datenabstraktion, Datenkapselung (ADT)

➥ **Objektorientierte Technologie (OOT)**



Beschränkungen der klassischen Datenmodelle

- **einfach strukturierte Datenobjekte**
  - satzorientiert, festes Format
  - nur einfache Datentypen
- **geringe semantische Ausdrucksfähigkeit**
  - fehlende Abstraktionskonzepte
  - begrenzte Auswahlmächtigkeit der Anfragesprachen
- **nur einfache Integritätsbedingungen**
- **umständliche Einbettung in Programmiersprachen**
- **auf kurze Transaktionen zugeschnitten (ACID)**
- **keine Unterstützung**
  - von Zeit und Versionen
  - von räumlichen Beziehungen
- **mangelnde Effizienz und Effektivität bei anspruchsvollen Anwendungen**
- . . .

➥ **Wie wirkt sich das bei komplexen Anwendungen aus?**

Erstes Beispiel: Landinformationssysteme

- **Problem:**
  Neue Anwendungen für relationale DBS zeigen Beschränkungen auf,
  z.B. Landinformationssysteme.
  (Sonderfall eines geographischen Informationssystems, GIS)
- **Zweck:**
  - Entscheidungshilfe in Verwaltung und Wirtschaft
  - Hilfsmittel für Planung und Entwicklung
- **Bestandteile:**
  - Datensammlung einer Region
  - Verfahren und Methoden zum Erfassen, Aktualisieren, Verarbeiten und Verbreiten der Daten
- **Grundlage:**
  - einheitliches räumliches Bezugssystem (Vermessungsfixpunkte)
  - 2D + Höhe für ausgezeichnete Punkte

  bestimmen die Eigenschaften der zu verwaltenden Daten

Landinformationssysteme (2)

- **Geographische Daten**
  - Die **räumlichen Daten** stehen im Mittelpunkt des gesamten Systems.
  - Für alle räumlichen Daten gibt es ein **einheitliches zweidimensionales Bezugssystem.**
  - Die **geometrische Information** ist in Vermessungspunkten und Grenzlinien enthalten.
  - Die **topologische Information** faßt Punkte und Linien zu Flächen, Kanten und Netzen zusammen.
  - Die **interaktive Arbeitsweise** erfordert einen schnellen Zugriff auf die räumlichen Daten.
  - Die räumlichen Daten können im **Raster-, Vektor- oder Hybridmodus** dargestellt werden.

- **Benutzersicht auf den zentralen Objekttyp FLURSTÜCK:**
  - in mehreren Registern geführt
  - identifizierende Nummer
  - liegen „dicht”, ohne „Löcher”, auch keine Überlappung
  - geschlossenes Polygon, beliebige Anzahl begrenzender Knoten
  - objektbezogener Zugriff (über Nummer, Besitzer usw.)
  - raumbezogener Zugriff (über Planquadrat)
  - nachbarschaftsbezogener Zugriff

Darstellung im Relationenmodell

- **Modellierungsansatz:**
  - *ein* Objekttyp für den Benutzerobjekttyp FLURSTÜCK
  - *eine* Ausprägung für jedes einzelne Flurstück:

```
FLURSTÜCK (F-NR,
           {Menge von Punkten o. Kanten}, ...)
```

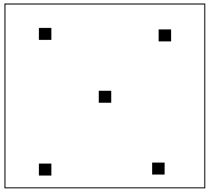
- **Normalisierung erforderlich!**
  Es sind keine mengenwertigen Attribute usw. erlaubt

- **Abbildungsvorschläge**

(1)

```
FLURSTÜCK (F-NR, ... )
F-PUNKTE (F-NR, LFD-NR, X-KOORD, Y-KOORD)
```

➥ LFD-NR sorgt für Eindeutigkeit:



+ Zugriff auf Flurstück: Verbund und ORDER BY
+ raumbezogener Zugriff: von den Punkten zu den Flurstücken
+ Nachbarschaft: ebenso
– Redundanz: Eckpunkte mehrfach
– Kanten nur implizit
– Konsistenz
– Änderungen

Darstellung im Relationenmodell (2)

(2)

```
FLURSTÜCK (F-NR, ... )
F-KANTEN (F-NR, K-NR, LFD-NR)
KANTE (K-NR, X1, Y1, X2, Y2, ... )
```

➥ jede Kante nur einmal dargestellt; (n:m)-Beziehung

(3)

analog, nur mit den Eckpunkten anstelle der Kanten,
(n:m)-Beziehung zwischen Flurstücken und Eckpunkten

(4)

```
FLURSTÜCK (F-NR, ... )
F-KANTEN (F-NR, K-NR, LFD-NR)
KANTE (K-NR, ... )
K-PUNKTE (K-NR, P-NR)
PUNKT (P-NR, X, Y, ... )
```

(5)

```
FLURSTÜCK (F-NR, ... )
KANTE (K-NR, F-NR1, F-NR2,
P-NR1, P-NR2, ... )
PUNKT (P-NR, X, Y, ... )
```


Darstellung im Relationenmodell (3)

- **Operationen des Datenmodells**
  z. B. Flurstück (Parzelle) lesen

  (1)

```
SELECT ...
FROM FLURSTÜCK F, F-PUNKTE P
WHERE F.F-NR = 999
  AND P.F-NR = 999;
```

  ➥ besser in zwei getrennten Anfragen;
  mühsame Berechnung der Kanten

  (4)

```
SELECT ...
FROM FLURSTÜCK F, F-KANTEN FK,
     KANTE K, K-PUNKTE KP, PUNKT P
WHERE F.F-NR = 999
  AND F.F-NR = FK.F-NR
  AND FK.K-NR = K.K-NR
  AND K.K-NR = KP.K-NR
  AND KP.P-NR = P.P-NR;
```

  ➥ vierfacher Verbund

- **Anwendungsnahe Operationen**
  - Einfügen Parzelle
  - Löschen Parzelle
  - Teilen Parzelle
  - Vereinigen zweier Parzellen

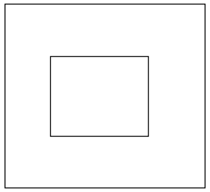
  ➥ Herunterbrechen auf SQL-Anweisungen

Darstellung im Relationenmodell (4)

- **Einhaltung der Konsistenzbedingungen**
  - jede Parzelle muß von einem geschlossenen Kantenzug begrenzt werden
  - es darf keine undefinierten Flächen (“Löcher”) zwischen/in Parzellen geben
  - Parzellen dürfen sich nicht überlappen

  ➥ als Prädikate bzw. Selektionsausdrücke (Assertion) in SQL formulieren???

- **Andererseits:**
  Kann



  modelliert werden?

- **Fazit**
  - Objekttyp Parzelle über diverse Relationen verstreut
  - die Relationen modellieren jeweils nur Teilaspekte des ursprünglichen Objekttyps
  - Integrität vom System kaum zu gewährleisten

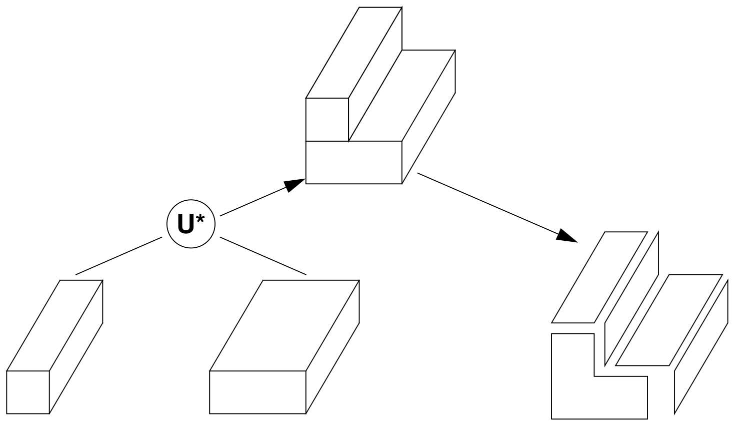
Zweites Beispiel: CAD-Systeme

- **Eigenschaften der Daten**
  - Technische Objekte sind i. allg. **dreidimensional.**
  - Die räumlichen Daten bilden nur einen kleinen, aber wichtigen Teil der Daten des Systems.
  - Die interaktive Arbeitsweise erfordert einen schnellen Zugriff auf die räumlichen Daten (Zeichnen am Bildschirm).
  - Jedes technischen Objekt besitzt sein eigenes Bezugssystem.
  - Die geometrische und topologische Information hängt vom gewählten Darstellungsschema ab.
  - Die räumlichen Daten können im Raster-, Vektor- oder Hybridmodus dargestellt werden.

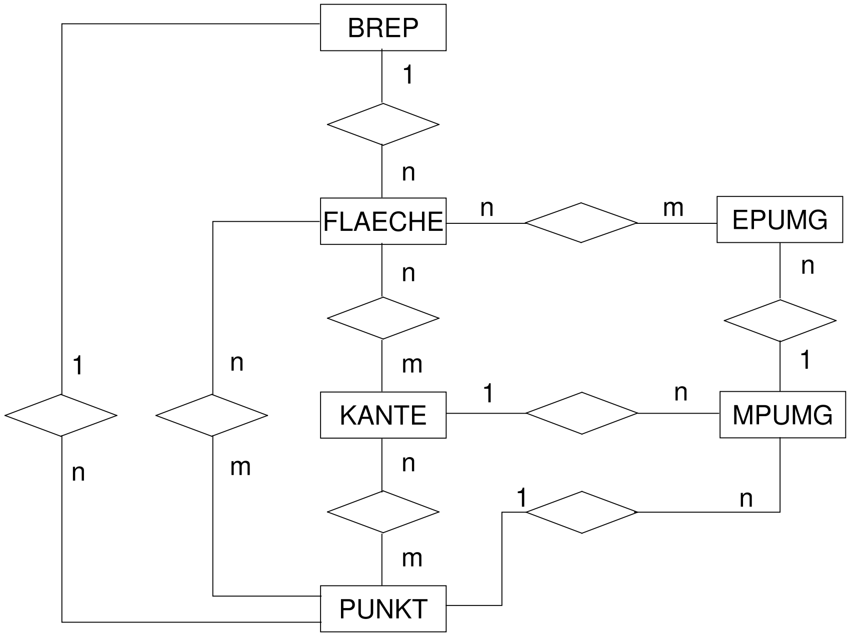
- **Geometrische Modellierung**
  - Konstruktionsprozeß für dreidimensionale Objekte (zusammengesetzte Körper)
  - Volumenorientierter geometrischer Modellierer (CSG = Constructive Solid Geometry)
  - Auswahl aus einer Menge von parametrisierten **primitiven Objekten**
  - Anwendung **regulärer Operatoren** (Vereinigung, Differenz, Translation, Rotation) zur schrittweisen Konstruktion komplexerer Objekte
  - System leitet automatisch **Begrenzungsflächendarstellung** ab (BREP = Boundary Representation)

CAD-Systeme (2)

- **CSG- und BREP-Modellierung**



- **DB-Repräsentation: Geometriemodell**
  als Entity-Relationship-Diagramm:



Darstellung im Relationenmodell

- **Abbildung**
  - wegen der vielen (n:m)-Beziehungen mehr als 10 Relationen!
  - Einzelteile des Körpers und ihre Beziehungen durch “unabhängige” Tupeln verschiedenen Typs dargestellt

  Schema:

```
BREP (BID, ...)
FLÄCHE (FID, ..., BID)
FK-KA (FID, KID)
KANTE (KID, K-LÄNGE, ...)
KA-PKT (KID, PID, ...)
PUNKT (PID, X, Y, Z, ...)
...
```

- **Operationen**

Anfrage: alle Punkte, die zum Bauteil 7853 gehören und Kanten mit einer Länge von mehr als 10 Einheiten begrenzen

```
SELECT PID, X, Y, Z
FROM PUNKT
WHERE PID IS IN

     (SELECT PID FROM KA-PKT
      WHERE KID IS IN

         (SELECT KID FROM KANTE
          WHERE K-LÄNGE > 10
          AND KID IS IN

             (SELECT KID FROM FK-KA
              WHERE FID IS IN

                (SELECT FID
                 FROM FLÄCHE
                 WHERE BID = 7853))));
```

Darstellung im Relationenmodell (2)

- **Operationen (2)**
  - Bohrung an einem Werkstück anbringen:
    „Subtraktion” eines Zylinders vom bisher konstruierten Körper

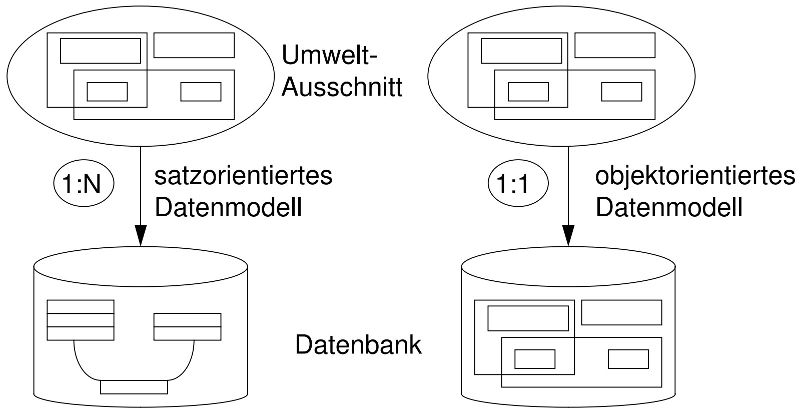
- **Fazit**
  - Komplexes Objekt durch heterogene Tupelmenge verkörpert.
    In vielfältiger Weise über Wertegleichheit von Attributen verknüpft
  - „Atomisierte” Sicht einzelner Tupeln
  - Verlorengegangen:
    ganzheitliche Sicht des 3D-Körpers und
    die Möglichkeit seiner integrierten Behandlung
    - Objektzugriff (Folge komplexer Verbundoperationen),
    - Kontrolle von Integritätszusicherungen,
    - Objektmanipulationen gemäß dem (semantisch weit höheren) Anwendungsmodell müssen mit den verfügbaren Operationen des Relationenmodells nachgebildet werden.

    ➥ Oft Tausende von Operationen!
  - Unnatürliche Präsentation/Ausgabe komplexer Objekte
    - Ergebnis ist riesige Tabelle (oder mehrere) mit enormer Redundanz – wo eine Sammlung verschiedener Tupel benötigt wird
    - Anwendung muß Beziehungen der Objektstruktur auswerten

    ➥ Strukturierte Ausgabe is im RM nicht möglich!

Objektorientierung bei DBS

- **Kernidee**



- **Speicherung und Suche von Objekten?**
- **Wirkung von Aktualisierungs-Operationen?**
  - Einfügen
  - Löschen
  - Kopieren, ...
- **Erhaltung der Konsistenz?**
- **Leistungsaspekte?**

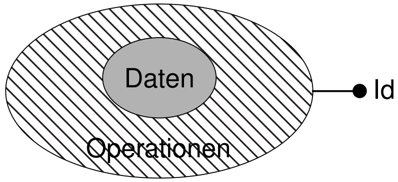
Definition eines objektorientierten DBS[1]

- **OODBS muß zwei Kriterien erfüllen**
  - Es muß ein DBS sein.
  - Es muß ein objektorientiertes System sein.
- **DBS-Aspekte:**
  - Persistenz (Dauerhaftigkeit von Daten über Programmausführung hinaus)
  - Sehr große Datenmengen (d. h. Zwang zur Verwendung von Externspeichern)
  - Synchronisation (d. h. Mehrbenutzerbetrieb)
  - Logging und Recovery (Datensicherung und -wiederherstellung)
  - Deskriptive Anfragesprache (Ad-hoc-Anfragen)
- **OOS-Aspekte:**
  Grundkonzepte der Objektorientierung
  - Objektidentität
  - Direkte Darstellung Komplexer Objekte
  - Datenkapselung
  - Typen oder Klassen, Typ-/Klassenhierarchien
  - Vererbung
  - Erweiterbarkeit (neue Typen, nicht unterscheidbar von systemdefinierten Typen)
  - Polymorphie: Überladen (overloading) und spätes Binden
  - Volle Berechenbarkeit (Mächtigkeit einer Programmiersprache)
- **Wahlweise Aspekte:**
  - Mehrfach-Vererbung, Typprüfung und -herleitung,
  - Verteilung, Lange Transaktionen

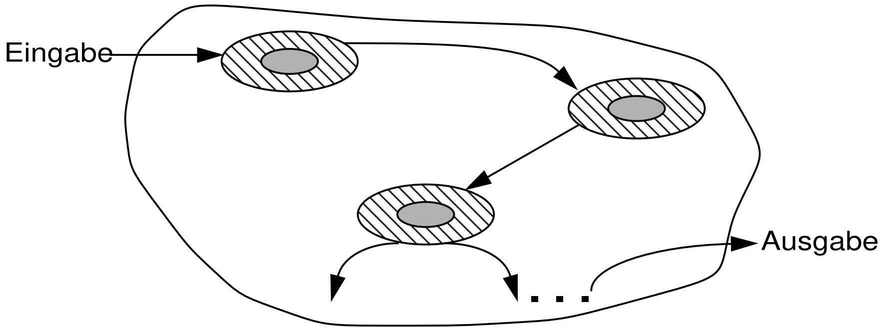
1. M. P. Atkinson, et. al: „The Object-Oriented Database System Manifesto", in: Won Kim, Jean-Marie Nicolas, and Shojiro Nishio (eds.), Proc. First Intl. Conf. on Deductive and Objekt-Oriented Databases, Elsevier Science Publishers, Amsterdam, 1989.

Fundamentale Idee

- **„Alles" ist ein Objekt?**



- **Objekteigenschaften**
  - Objekte haben einen Identifikator
  - Objekte haben einen internen Zustand, beschrieben durch Attribute (Variable, Slots, . . .)
  - Objekte haben eine Schnittstelle zur externen Welt, definiert durch die Menge an Operationen
  - Objekte kommunizieren über Nachrichten



- **Verarbeitungsaspekte**
  - Suchen/Aktualisieren durch Methodenaufrufe
  - Integritätskontrolle
  - Autorisierung/Zugriffskontrolle

➥ **Ist diese Sichtweise bei DBS angemessen?**
(Deskriptivität, Mengenorientierung, Wertbezug usw.)

Objektidentität

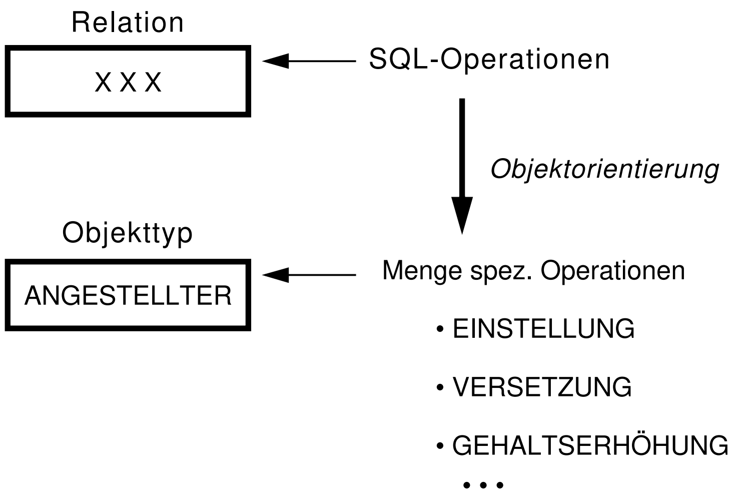
- **Objektidentität**
  - keine anwendungsspezifischen Werte (wie im RM)
  - Identitätskonzept des Relationenmodells zu schwach (identity thru contents)
  - sondern durch eindeutige Objekt-Identifikatoren (Surrogate)
- **Objekt-Identifikatoren (OIDs, Surrogate)**
  - tragen keine Semantik (↔ Primärschlüssel im RM)
  - während der Objektlebensdauer konstant
  - üblicherweise systemverwaltet
- **Eigenschaften/Konsequenzen**
  - Existenz des Objektes ist unabhängig von seinem Objektzustand
    - Änderungen beliebiger Art (auch des Primärschlüssels im RM) ergeben *dasselbe* Objekt
  - Identität # Gleichheit (beides ist ausdrückbar)
    - Objekte können *identisch* (dasselbe Objekt) oder *gleich* (derselbe Wert) sein
  - OID zur Darstellung von Referenzen/Beziehungen
    - Realisierung gemeinsamer Teilobjekte ohne Redundanz möglich (referential sharing)
    - keine irreführenden Referenzen auf Objekte

Komplexe (strukturierte) Objekte

- **Anwendung von Typ-Konstruktoren -**
  Wünschenswerte Konstruktoren:
  - ARRAY-Konstruktor (VECTOR)
  - RECORD / TUPLE
  - LIST / SEQUENCE
  - SET
  - MULTISET / BAG
- **Eigenschaften:**
  - Orthogonalität der Konstruktoren
  - beliebige (rekursive) Kombination von Konstruktoren zum Aufbau komplex strukturierter Objekte
  - Operationen zur Verarbeitung der (beliebig) strukturierten Objekte
- **Ein OODBS sollte wenigstens unterstützen:**
  - die Typkonstruktoren TUPLE, LIST und SET und
  - ihre beliebige Kombination

ADTs / Kapselung - Beispiele

- **Objektebene**
  - Unterschied zum RM



- **Attributebene:** Erzeugung problembezogener Datentypen mit zugeschnittenen Operatoren und Funktionen

Beispiel: ADT 'DATE' , Operator '–'

	Normalfall	Finanzwelt
15. April – 15. März	31	30
15. März – 15. Febr.	28	30

```
SELECT   'Beschäftigungsdauer:', HEUTE () – P.EDATUM
FROM     PERS P
```

- **erhöhte Datenunabhängigkeit**
- **Verwaltung der Funktionen im DBS (stored procedures)**

  ➥ reduzierter Kommunikationsaufwand mit DBS

ODMG-Objektmodell: Überblick

- **ODMG-Standardisierung (ODMG93) besteht aus**
  - Objektmodell
  - Objekt-Definitionssprache (ODL)
  - Objekt-Anfragesprache (OQL)
  - Sprachbindungen (language bindings) für C++ und Smalltalk
- **Objektmodell:**
  Unterscheidung zwischen Objekten und Literalen
  - Literale sind nicht änderbar und haben keinen Objekt-Identifier
  - Objekte sind änderbar und haben OID

    ➥ sowohl Literale als auch Objekte haben einen Typ
- **Literale sind entweder atomar, strukturiert oder von einem Kollektionstyp**
  - vordefinierte atomare Literaltypen:
    Long, Short, Boolean, Char, String, Enum, ...
  - strukturierte Literaltypen; vordefiniert (Date, Time, Timestamp, Interval) oder benutzerdefiniert
  - Kollektionstypen: Set, Bag, List, Array
- **Objekte sind entweder atomar oder von einem Kollektionstyp**
  - atomare Objekte sind stets benutzerdefiniert
    (keine vordefinierten atomaren Objekttypen)
  - Kollektionstypen analog zu Literalen
- **Objekte eines Typs weisen gemeinsame Charakteristika bezüglich Zustand und Verhalten auf**
  - Zustand: Werte für bestimmte Eigenschaften (properties)

    a) Attribute

    b) (symmetrische) Beziehungen zu anderen Objekten
  - Verhalten: Menge von Operationen

Typen

- **Typen bestehen aus**
  - Schnittstelle (Interface): Festlegung der sichtbaren Eigenschaften und Operationen
  - Implementierung: Festlegung der Datenstrukturen und Methoden zur Realisierung der Schnittstelle
- **Klasse: Kombination einer Typ-Schnittstellenspezifikation und einer Implementierung**
- **Implementierung ist sprachabhängig und im DB-Schema nicht sichtbar**
- **Beschreibung der Schnittstelle durch ODL bzw. in bestimmter Programmiersprache**

```
interface Person {
        attribute string Name;
        attribute date Geburtsdatum;
        attribute enum Geschlecht {m, w};
        integer Alter ();
    }
```

- **Typen können durch Sub-Typen spezialisiert werden**
  - Definition zusätzlicher Attribute, Beziehungen (Relationships) und Operationen (Methoden)
    z. B.: Professor, Student, Prüfling
  - Einfach- und Mehrfachvererbung
  - Operationen können überladen werden (Overloading)

OQL (Object Query Language)

- **Spezifikation von Ad-Hoc-Anfragen sowie von eingebetteten Anfragen**
- **Eigenschaften der OQL**
  - keine vollständige DML bzw. OML
  - keine Änderungsoperationen (Änderungen müssen durch typspezifische Operationen erfolgen)
  - Verarbeitung beliebiger Objekte
    (nicht nur von Tupeln und Tabellen/Sets)
  - Unterstützung von Pfadausdrücken sowie von Methodenaufrufen innerhalb von Anfragen
  - OQL-Aufrufe aus Anwendungsprogrammen möglich
    (für Programmiersprachen mit ODMG-Binding)
  - deklarativer Zugriff
  - an SQL angelehnt (insbesondere seit Revision 1.2)
- **funktionale Anfragesprache**
  - Query = Ausdruck
  - orthogonale Verwendbarkeit von Anfrageausdrücken
- **Beispiel:**

  select x.Matnr
  from Student x
  where x.Name = ‘Schmidt’

OQL (2)

- **Anfrageergebnis**
  - ist entweder Kollektion von Objekten, Kollektion von Literalen, ein Objekt oder ein Literal
  - kann explizit strukturiert werden

    select distinct x.Alter
    from Professor x
    where x.Name = ‘Meier’

    select struct (A: x.Alter, P: x.PNR)
    from Professor x
    where x.Name = ‘Meier’

    select struct (Prof: x.Name, sgS: select y from x.Prüfling y
    where y.Note < 1,5))
    from Professor x

- **direkte Traversierung entlang von Relationships anstelle von Joins**
  - Zugriffe über ‘’.’’ oder ‘’->’’ - Notation
  - Voraussetzung: keine der Referenzen weist Nullwert (nil) auf

    p.lebt-in.Haus.Adresse.Straße

- **Verwendung von Methodenaufrufen wie Attributzugriff möglich**
  - kein syntaktischer Unterschied bei parameterlosen Methoden
  - Methode kann komplexen Wert liefern
    -> Verwendung innerhalb von Pfadausdrücken möglich

    select max(select k.Alter from p.Kinder k)
    from Person p
    where p.Name = ‘Meier’

Objektorientierte Datenbanksysteme

- **Vorsicht: verschiedene, teils widersprüchliche Definitionen!**

- **Beispiele:**

  sinngemäß: „Ein OODBS zeichnet sich dadurch aus, daß jedes beliebig komplexe Objekt des Diskursbereichs durch genau ein (ebenfalls komplexes) Datenbankobjekt dargestellt werden kann.”

  „Ein OODBS ist ein objektorientiertes System und ein Datenbanksystem.”

  ➥ umfaßt also eine Programmiersprache!

  „Ein OODBS ist ein DBS mit einem objektorientierten Datenmodell.”

- **Das Paradigma der Objektorientierung in der DB-Technologie**

  Besonderheit in der **Datenbankwelt** (und Ursache vieler Mißverständnisse):
  Ausdehnung des Begriffs Objektorientierung
  auf die Konzepte zur Verwaltung **komplexer Objekte**!

  → keine Methoden, keine Einkapselung, **. . .**

  → „strukturelle” Objektorientierung (PRIMA, DAMOKLES, **. . .**)

  ➥ Aber Vorsicht:
  Ansätze zur Darstellung der Teil-Ganze-Beziehung nicht besonders ausgereift (ER-Modell und wenig mehr)

- **Im Bereich der Programmiersprachen**
  meint Objektorientierung dagegen meist Smalltalk und Nachfolger, also:
  - Klassen mit Methoden, Klassenhierarchie, Vererbung, Überladen, Polymorphismus, **. . .**
  - sämtliche Attributwerte sind Verweise (genauer: Objektidentifikatoren)
  - Teil-Ganze-Beziehung ist praktisch kein Thema

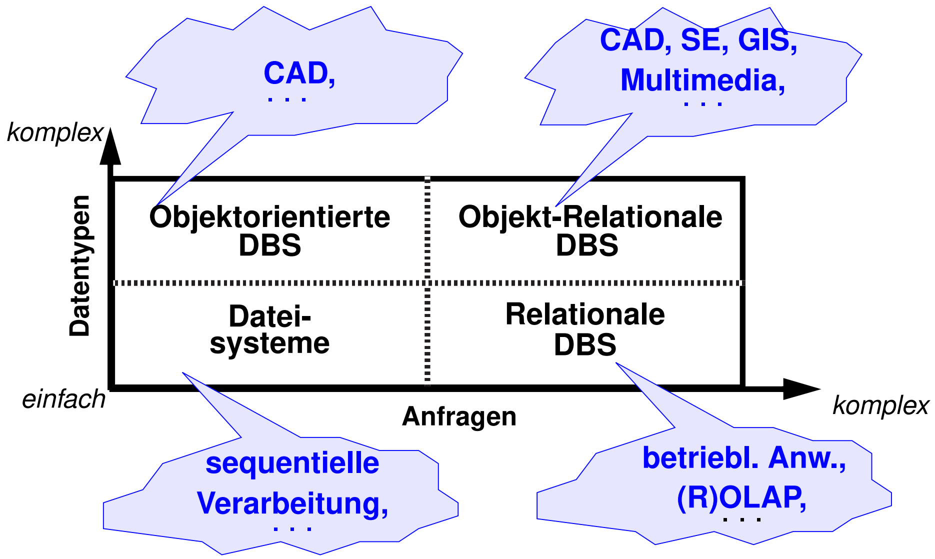
Vergleich von Begriffen/Konzepten

OR

relational ⟷ objektorientiert (C++, Java, Smalltalk)

	relational	objektorientiert
Intensional:	• Relationenschema	• Objekttyp
Extensional:	• Relation	• Klasse/Kollektion
Struktur:	• sichtbar im Schema	• unsichtbar: gekapselt, Signatur
Sprache:	• generische Operationen („Insert into Pers”)	• typspez. Operationen („Einstelle Angestellter“)
Identität:	• wertbasiert (Primärschlüssel)	• objektbasiert (OID)
Zugriff:	• mengenorientiert (n Tupel)	• satzorientiert (1 Objekt)
	• deskriptiv (Anfragen über n Relationen)	• navigierend (Iterator mit Suchargument)
	• n-mengenorientiert	• 1-mengenorientiert

Objekt-Relationale DBS – Vision



- **Erwünschte Eigenschaften von *Objekt-Relationalen DBS (ORDBS)***
  - **Eigenschaften von RDBS**
    - \+ ADTs/Kapselung
    - \+ Klassen, Vererbung
    - \+ mengenwertige Attribute, OIDs/Referenzen
    - \+ benutzerdefinierte Funktionen
    - \+ navigierende, prozedurale Verarbeitung
    - \+ Multimedia-Integration
    - \+ Erweiterbares Typsystem und Erweiterungsinfrastruktur
    - \+ Client/Server-Verarbeitung
    - \+ Offenheit
    - \+ . . . ?
  - **Integration**
    (Leistungsverhalten, Skalierbarkeit, Bereitstellung auf Client) ?

Objekt-Relationale DB-Konzepte: Motivation

- **Relationale Datenbankverwaltungssysteme bieten**
  - eine Menge von Datentypen, um Anwendungsdaten darstellen zu können
  - eine Menge von Operationen, um diese Datentypen manipulieren zu können

- **Beispiele:**

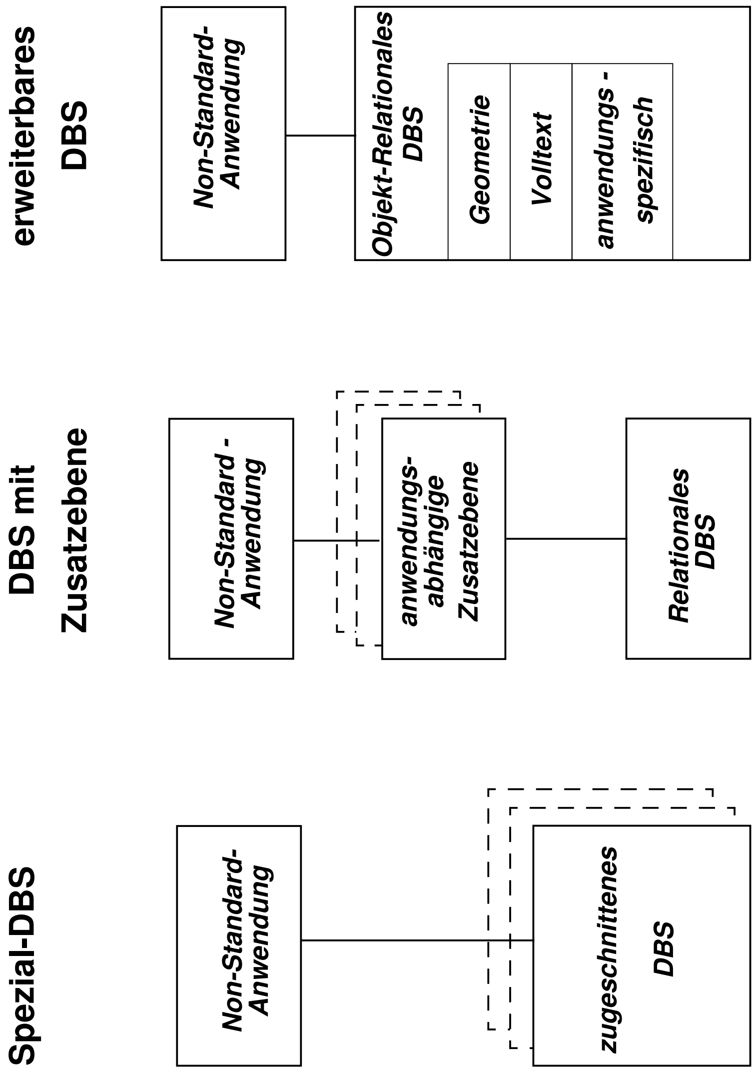
Datentypen	Funktionen
INTEGER	+, - , *, /, AVG, SUM, ...
CHARACTER	Manipulation von Zeichenketten: suchen, anfügen, ...
DATE	Tag, Monat, Jahr, +, -, ...

- **Neue Anwendungen erfordern neue Datentypen und Funktionen!**

- **Beispiele:**

Datentypen	Funktionen
TEXT	Volltextsuche, Rechtschreibkorrektur, ...
POLYGON	Durchmesser, Schnitt von Polygonen, Fläche, ...
RASTER	Konvertierung zwischen Formaten, Farbanalyse

Drei verschiedene System-Architekturen



Objekt-Relationale Datenbank-Technologie

- **Funktionalität wird derzeit im wesentlichen durch den Standard SQL99 beschrieben**
- **Erhöhung der Anfragemächtigkeit**
  - Allgemeine Tabellenausdrücke
  - Rekursion
  - Große Objekte
- **Unterstützung von benutzerdefinierten Typen (UDT) bzw. Objektorientierung**
  - komplexe Datenstrukturen mit
  - komplexer Funktionalität definierbar
  - Vererbungshierarchie
  - . . .

  ➥ Repräsentation von Anwendungswissen im DB-System (Klassen-Bibliotheken)
- **Erweiterung von herkömmlichen Tabellen**
  - komplexe Spalten (Attribute, Wertebereiche)
  - Schachtelung
  - Referenzierung/Dereferenzierung
  - Tabellen mit Typbindung (typed tables) und Tabellenhierarchien
  - . . .
- **Erweiterungsinfrastruktur**
  - benutzerdefinierte Datentypen und Funktionen lassen sich in das ORDBS integrieren und sind in SQL nutzbar
  - Unterstützung durch spezielle Zugriffspfade und Speicherungsstrukturen
  - Integration mit DBS-Komponenten wie Optimizer, Synchronisation, Logging und Recovery

Zusammenfassung

- **OODM liefern leistungsfähige Konzepte für den Umgang mit komplexen Objekten und mächtigen Operationen**
  - Sie eignen sich für Non-Standard-Anwendungen
  - Es gibt bereits leistungsfähige Implementierungen von OODBS
- **OO-Manifesto ist nicht allgemein anerkannt**
  - Wieviele Eigenschaften sind essentiell?
  - Welche Eigenschaften sind eher ergänzend?

    ➥ Es werden noch viele weitere Forderungen gestellt!
- **Es gibt nicht ein OODM (DDL und DML), sondern viele!**
  - Rolle von ODMG ist nicht ausreichend akzeptiert:
    Sie hat eine OODM-Standardisierung mit einer mengenorientierten Anfragesprache ODMG-OQL erarbeitet
  - Reichhaltige Strukturierungsmöglichkeiten bei der Festlegung des DB-Schemas
  - Systemauswahl bleibt trotzdem schwierig
- **Es gibt ein durch SQL99 standardisiertes ORDM**
  - Es wurden die wesentlichen OODM-Konzepte übernommen
  - Typkonstruktoren, benutzerdefinierte Typen und Funktionen
  - Typ- und Tabellenhierarchien sowie Referenzen
  - Regelsystem (Triggerkonzept)
  - Erweiterungsinfrastruktur
  - . . .