Prüfungsfragen

Prof. Dr. Meyer-Wegener, kein Beisitzer
Unbenoteter Schein
Angenehme Atmosphäre bei einem Glas Wasser; es kam ihm darauf an, dass ich mich „mit allem mal beschäftigt“ habe.

Beschreiben Sie eins der beiden älteren Datenmodelle
hierarchisch oder CODASYL: Typen und Ausprägungen sind Baumstruktur...
Vorteile/Nachteile/Probleme
Keine Prüfbarkeit der Zuordnungen, nur 1:N-Beziehungen
Welche Operationen gibt es in diesem Modell?
Satzorientiert: Finden, navigieren, lesen, einfügen unter/nach aktuellem Satz...

Warum brauchte man objektorientierte Datenbanksysteme?
Speicherung komplexer Objekte, Beispiel: Flurstück mit Flächen, Kanten, Operationen, Integrität
Wie würde man das Flurstück mit Relationen darstellen?
Flurstück(FS-ID), Kante(K-ID, FS-ID), Punkt(P-ID, K-ID, X, Y)
Welche Probleme treten in dieser Darstellung auf?
Zugriffe über etliche Joins, Änderungen wie Aufteilen nur mit komplexer Anwendungslogik, keine Integritätsprüfung

Welche andere Form der Objektorientierung gibt es in OODB?
Integration in OO-Programmiersprachen: direktes Abspeichern der Anwendungsobjekte in der DB, kein Impedance Mismatch

Zurück zur strukturellen Objektorientierung: Wie modelliert man das Flurstück dort?
Zusammengesetzte/mehrwertige Attribute (NF^2)
Probleme: Redundanz der Daten; keine Anwendungsneutralität, da verschachtelte Relationen
Irgendwas zum ODMG-Modell...
An OMG angelehnt, besteht aus ODL (nach OMG-IDL), OQL, Programmiersprachenintegration
ODL beschreibt Objekte mit Typhierarchie, Attributen, Methoden und Beziehungen (wichtig!)
Beziehungen sind bidirektional, DBS gewähleistet referenzielle Integrität
Beziehungen können schneller ausgewertet werden als Joins über Werte
Wie sieht das Flurstück in ODL aus?
interface Flurstück { relationship Set<Kante> Kanten inverse Kante.Flurstück }
interface Kante { relationship Flurstück Flurstück inverse Flurstück.Kanten, relationship Set<Punkt> Punkte inverse ... }
interface Punkt { relationship Kante Kante inverse Kante.Punkte, ... }
Löst das unsere o.g. Probleme?
Redundanz: N:M-Beziehungen können sehr einfach als Set eingefügt werden: interface Kante { relationship Set<Flurstück> Flurstücke inverse... }
Anwendungsneutralität: Daten können auch „von unten“ gelesen werden, dank bidirektionaler Beziehungen
Änderungen/Integrität: Nicht direkt durch DBS gegeben, kann aber in Methoden gekapselt werden, die von allen Anwendungen verwendet werden