1 Grundkonzepte der datenorientierten Modellierung

1.1 Ein einführendes Beispiel

1.1-1/3: Beim letzten Item *Abteilungen* gilt (wie sich bei den Beispielen herausstellt), daß auch mehrere Abteilungen in einem Gebäude untergebracht sein können.

1.2 Konventioneller Ansatz

1.2.1 Info'system in COBOL mit Dateien

1.2.1-1/2: Offenbar ist PersNr nur innerhalb des Betriebes eindeutig. Auch die GebNr ist nur innerhalb des Standorts eindeutig.

1.2.2 Änderung des Dateiaufbaus

1.2.2-2/2: Hier wird gesagt, daß "Änderung des Dateiaufbaus die Änderung der Dateibeschreibung, damit Änderung und Neuinstallation aller Programme, die auf diese Datei zugreifen, zur Folge hat." Ich frage mich, *ob sich das durch das DB-Konzept wirklich vermeiden läßt, und wenn ja, wie (konkret)?*

1) Verlängern oder Verkürzen eines Feldes.
   Beispiel: Seit 1.7.1993 gab es neue Postleitzahlen.
2) Entfernen oder Hinzufügen eines Feldes.
   Beispiel: In die PERSONAL-Datei soll für jeden Mitarbheiter das Geburtsdatum aufgenommen werden.
3) Verändern der Anordnung der Felder.

1.2.3 Redundanz

1.2.4 Inkonsistenzen

1.2.5 Beurteilung des konventionellen Ansatzes

Dazu gibt es die Folien **1.2.5-1/2: Insel-Lösungen** und **1.2.5-2/2: Programmierung im konventionellen Stil**. Gefällt mir nicht so besonders. Ich würde hier deutlicher auf die *Datenabhängigkeit* (Stucky: *Programm-Daten-Abhängigkeit*) als zentralen Begriff und "Quelle aller Übel" fokussieren. "Insellösungen mit privaten (und redundanten) Dateien der Anwendungen" und überhaupt die "Programmierung im konventionellen Stil" sind nur Konsequenzen der Datenabhängigkeit.

1.3 Datenbank-orientierter Ansatz

1.3.1 Das Konzept

1.3.1-1/7: Auch hier würde ich in Entsprechung zu **1.2.5** die *Datenunabhängigkeit* (data independence) als zentralen Begriff und Lösungsansatz in den Mittelpunkt stellen.

1.3.1-3/7: Abgesehen von den weggefallenen Feld- und Satzlängenangaben unterscheidet sich die PERSONAL-Tabelle nicht von der PERSONAL-Datei. Verdeutlicht dem Studenten also nicht unbedingt den durch das DB-Konzept realisierten "Quantensprung".

Die in angesprochenen **1.3.1-4/7** "Geschäftsregeln" sind wohl die in **1.1-1/3** aufgezählten Punkte. Diese beziehen sich aber auf die beiden Items (–) auf der Folie **1.3.1-5/7**.

1.3.1-7/7 Ich mache bei den Integritätsbedingungen im relationalen Modell gerne die Unterscheidung: *modell-immanente* versus *benutzerdefinierte* Integritätsbedingungen. Die modell-immanenten sind: *Entitäts-* und *referentielle Integrität*. Die *benutzerdefinierten* sind wohl mit den hier angesprochenen *semantischen* Integritätsbedingungen gleichzusetzen. Es fragt sich also, ob wir es bei diesem Beispiel in meiner Terminologie mit benutzerdefinierter oder modell-immanenter Integritätsbedingung zu tun haben. [Ist aber in diesem einführenden Stadium letztlich Haarspalterei]

1.3.2 Beurteilung des Datenbankansatzes

1.3.2-1/1 Kann man vielleicht auch etwas "schlüssiger" machen. Auf jeden Fall würde ich Item 1 mit der Überschrift *Datenunabhängigkeit der Anwendungen* versehen. [Müßte man eigentlich etwas relativieren]

1.4 Probleme beim Tabellenentwurf

1.4.1 Anomalien

[Betreffen genauso den Dateientwurf]

1.4.2 Zerlegung von Tabellen

1.5 Die Join-Operation

Ich deutsche das im Buch als "Verbund" ein. Ebenso etwa *Vetter, Zehnder*.

1.5-1/7: Für eine *gemeinsame Spalte* (gemeinsames Attribut) genügt nicht bloß ein *gemeinsamer Name*. Es muß auch ein *gemeinsamer* (oder wenigstens kompatibler) *Wertevorrat* (Domäne) gegeben sein.

1.6 Architektur eines Datenbanksystems

1.6.1 Grundfunktionen eines DBMS

1.6.2 Drei-Ebenen-Architektur

1.7 Datenmodelle als Beschreibungsmittel

Ich mag nicht diese undifferenzierte Verwendung der Bezeichnung *Datenmodell*. Ich unterscheide zwischen *logischen* (oder auch *abstrakten*) Datenmodellen und *semantischen* Datenmodellen. Die ersteren sind eine (high-level) abstrakte Programmiersprache, die Beschreibungsmittel für die strukturellen (Relation, Primärschlüssel, Fremdschlüssel, Integritätsbedingung) und manipulativen Aspekte (Relationenalgebra, Relationenkalkül, relationale Wertzuweisung) der Daten bereitstellt. Die letzteren stellen Beschreibungsmittel bereit, um ein konkretes Datensystem (als Repräsentation eines interessierenden Umweltausschnittes, soweit das für die jeweiligen Zwecke relevant ist) zu beschreiben und die dabei bestehenden Zusammenhänge, Regeln, Gesetzmäßigkeiten zu erfassen.

1.7.1 Realwelt und Modell

Hier geht es um semantische Modellierung.

1.7.2 Die klassischen Datenmodelle

Hier geht es um die "klassischen" logischen (oder abstrakten) Datenmodelle.

1.7.3 Semantische Datenmodelle

Ich würde sagen, hier geht es um die Abbildung eines semantischen Datenmodells (zb ERM) auf ein logisches Datenmodell (zb relationales) und in weiterer Folge auf eine konkrete Datenbanksprache, wie sie in einem bestimmten DBMS implementiert ist.

1.7.3-1/2: Hier wird neben dem *Entity-Relationship-Modell* (ERM) auch das *Semantisch-Hierarchische Modell* (SHM) angeführt.
Das ERM stammt übrigens von Chen, P.P.S.: The Entity Relationship Model: Towards a Unified View of Data, *ACM Transactions on Database Systems*, vol.1, 1976.

Dann gibt es noch das 'Extended Relational Model' (RM/T): Codd, E.F.: Extending the Database Relational Model to Capture More Meaning, *ACM Transactions on Database Systems*, vol.4, no. 4 (December 1979). [vgl. Date 6th ed, 349, 367, 565]
Das *Semantisch-Hierarchische Modell* (SHM) basiert auf dem paper: Lausen, G., Schek, H.-J.: Semantic Specification of Complex Objects. in: Proceedings IEEE Office Automation Symposium. Gaithersburgh 1987.