Zusammenfassung: Jahr 2

Inhaltsverzeichnis

1 Lernfeld 6 - Datenbanken / OENI

1.1 Codd'sche Regeln - Grundlagen des Relationenmodells

Edgar F. Codd beschrieb in den 1980'ern dreizehn Regeln, nummeriert 0 bis 12, die als Grundlage des Relationenmodells dienen. Ein Datenbank Managment System (DBMS), dass von sich behauptet, relational zu arbeiten, muss sich an diese Regeln halten.

0. The Foundation rule: For any system that is advertised as, or claimed to be, a relational data base. management system, that system must be able to manage data bases entirely through its relational capabilities.

1. The information rule: All information in a relational data base is represented explicitly at the logical level and in exactly one way — by values in tables.

2. The guaranteed access rule: Each and every datum (atomic value) in a relational data base is guaranteed to be logically accessible by resorting to a combination of table name, primary key value and column name.

3. Systematic treatment of null values: Null values (distinct from the empty character string or a string of blank characters and distinct from zero or any other number) are supported in fully relational DBMS for representing missing information and inapplicable information in a systematic way, independent of data type.

4. Dynamic online catalog based on the relational model: The data base description is represented at the logical level in the same way as ordinary data, so that authorized users can apply the same relational language to its interrogation as they apply to the regular data.

5. The comprehensive data sublanguage rule: A relational system may support several languages and various modes of terminal use (for example, the fill-in-the-blanks mode). However, there must be at least one language whose statements are expressible, per some well-defined syntax, as character strings and that is comprehensive in supporting all of the following items:

    - Data definition.
    - View definition.
    - Data manipulation (interactive and by program).
    - Integrity constraints.
    - Authorization.
    - Transaction boundaries (begin, commit and rollback).

6. The view updating rule: All views that are theoretically updatable are also updatable by the system.

7. High-level insert, update, and delete: The capability of handling a base relation or a derived relation as a single operand applies not only to the retrieval of data but also to the insertion, update and deletion of data.

8. Physical data independence: Application programs and terminal activities remain logically unimpared whenever any changes are made in either storage representations or access methods.

9. Logical data independence: Application programs and terminal activites remain logically unimpared when information-preserving changes of any kind that theoretically permit unimpairment are made to the base tables.

10. Integrity independence: Integrity constraints specific to a particular relational data base must be definable in the relational data sublanguage and storable in the catalog, not in the application programs.

11. Distribution independence: A relational DBMS has distribution independence.

12. The nonsubversion rule: If a relational system has a low-level (single-record-at-a-time) language, that low level cannot be used to subvert or bypass the integrity rules and constraints expressed in the higher level relational language (multiple-records-at-a-time).

1.2 Entwurf einer Datenbank

Beim Entwurf einer Datenbank werden vier Schritte unterschieden. Erstens muss anhand des Zwecks respektive des Kundenwunsches eine Zweckbestimmung durchgeführt werden. Daraus ergibt sich die Informationsstruktur. Anschließend lässt sich diese im semantischen Modell als *Entity Relationship Model* (ERM) darstellen. Im nächsten Schritt wird dieses Modell in ein Relationenmodell übersetzt. Abschließend wird mit der Implementierung ein physisches Modell erstellt. Erst die Implementierung ist von einem konkreten DBMS abhängig.

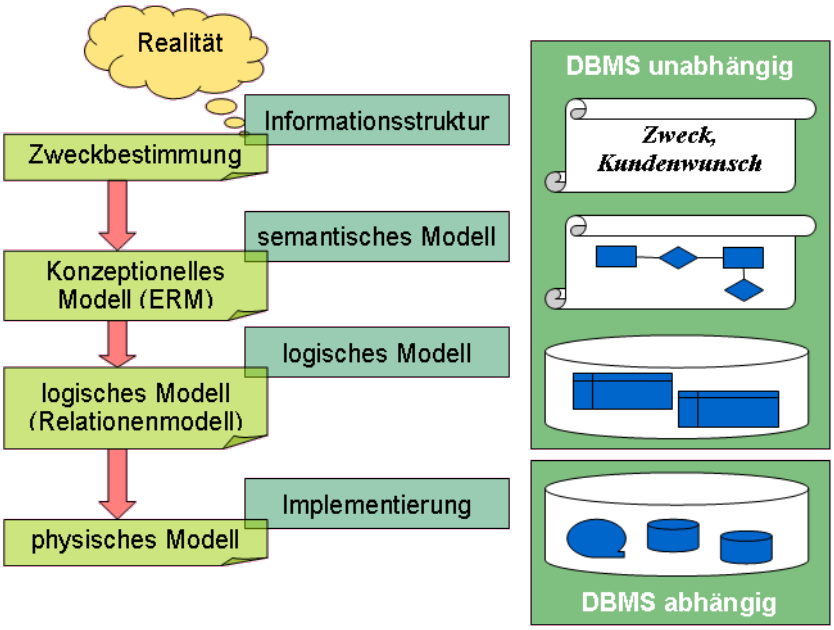

Abbildung 1: Phasen der Datenbankentwicklung

1.2.1 Informationsstruktur - Zweckbestimmung

1.2.2 Semantisches Modell - ERM

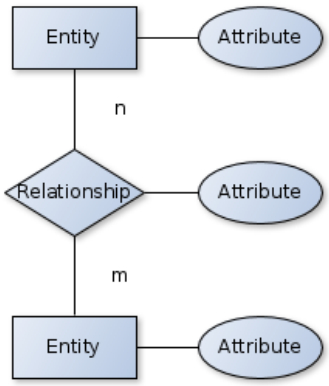

Abbildung 2: Basics des ERM

Bei dem Entity Relationship Model nach Peter Chen gibt es genau definierte Elemente. Dazu gehören drei Formen, das Rechteck, die Elipse und die Raute. Im nebenstehenden Bild ist zu erkennen, welche Form für welches Element steht. 1. Das Rechteck beschreibt Entitätstypen, wobei eine einzelne Entität ähnlich einem Objekt eine Instanz eines Entitätstypes ist. 2. Die Elipse beschreibt Attribute, also Eigenschaften, die ein Entitätstyp haben kann. Beispiele dazu folgen. 3. Die Raute beschreibt die Relationen zwischen verschiedenen Entitätstypen. An den Relationen können drei verschiedene Kardinalitäten die Art der Relation näher bestimmten; a) 1 zu 1, b) 1 zu n und c) n zu m. Nur bei n zu m Kardinalitäten können Attribute wie im Bild zu sehen mit eigenen Attributen ausgestattet werden. **Beispiele** für Kardinalitäten. Die n zu m Kardinalität könnte durch ein Attribut wie beispielsweise „Dauer“ ergänzt werden.

- 1 zu 1: Mitarbeiter hat Name
- 1 zu n: Mitarbeiter arbeitet in Abteilung
- n zu m: Mitarbeiter arbeitet an Projekt

1.2.3 Logisches Modell - Relationenmodell

Im Anschluss an die Erstellung eines ERM wird das Relationenmodell erstellt. Dabei gilt, dass jeder Entitätstyp einer Tabelle entspricht und jedes Attribut einer Spalte. Die Relationen werden durch Primär- und Fremdschlüssel dargestellt. Bei einer 1 zu 1 Relation

1.2.4 Physisches Modell - Implementierung

1.2.5 Joins - Grundtypen

Es wird zwischen zwei Grundtypen von Joins unterschieden, OUTER und INNER Joins. OUTER Joins können nochmals in drei Grundtypen unterschieden werden, sodass sich insgesamt vier Grundtypen von Joins ergeben: (1) FULL OUTER, (2) LEFT OUTER, (3) RIGHT OUTER und (4) INNER.

Beispiel - A und B

- SELECT ... FROM A FULL OUTER JOIN B ON A.id = B.id Der Befehl gibt sowohl A als auch B aus.

Beispiele - inklusive / exklusive Schnittmenge

- SELECT ... FROM A *LEFT* OUTER JOIN B ON A.id = B.id Der Befehl gibt A **inklusive** der Schnittmenge von A und B aus.
- SELECT ... FROM A LEFT OUTER JOIN B ON A.id = B.id WHERE B.id IS NULL Der Befehl gibt A **exklusive** der Schnittmenge von A und B aus.
- SELECT ... FROM A *RIGHT* OUTER JOIN B ON A.id = B.id Der Befehl gibt B **inklusive** der Schnittmenge von A und B aus.
- SELECT ... FROM A RIGHT OUTER JOIN B ON A.id = B.id WHERE A.id IS NULL Der Befehl gibt B **exklusive** der Schnittmenge von A und B aus.

Beispiele - Schnittmenge und Komplement

- SELECT ... FROM A *INNER* JOIN B ON A.id = B.id Der Befehl gibt die Schnittmenge von A und B aus.
- SELECT ... FROM A FULL OUTER JOIN B ON A.id = B.id WHERE A.id IS NULL OR B.id IS NULL Der Befehl gibt das Komplement der Schnittmenge aus - also alles, was sich A und B **nicht** teilen.

2 Eine Datenbank entwickeln

2.1 Normalisierung

Durch Normalisierung sollen Anomalien und Redundanz in einer Datenbank reduziert werden. Dabei erhöhen sich aber auch die Anforderungen an die Performance. Es werden drei Anomalien unterschieden: (1) Einfüge-Anomalien, (2) Lösch-Anomalien und (3) Änderungs-Anomalie.

(1) Einfüge-Anomalie: neue Daten können nicht angelegt werden, wenn noch nicht alle Informationen bekannt sind. (2) Lösch-Anomalie: werden Datensätze gelöscht, gehen mehr Informationen verloren, als eigentlich beabsichtigt war. (3) Änderungs-Anomalie: führen häufig zu Inkonsistenzen

Erste Normalform: Die Daten in einer Tabelle müssen atomar (unteilbar) sein und es darf keine Aufzählung (Wiederholungsgruppen) geben. Nach der Übertragung in die erste Normalform (1. NF) kann es notwendig werden, einen neuen (zusammensetzten) Primary Key zu definieren. **Zweite Normalform**: Die Tabellen muss in der 1. NF sein und jedes Nicht-Schlüsselattribut muss voll funktional vom gesamten Primärschlüssel abhängig sein. Voll funktional bedeutet in diesem

Zusammenhang, dass es zum Wert des Primärschlüssels genau einen Wert aus der jeweiligen Spalte des Nicht-Schlüsselattributes gibt. **Dritte Normalform**: Die tabelle muss sich in der 2. NF befinden und kein Nicht-Schlüsselattribut transitiv vom Primärschlüssel abhängt. **Boyce-Codd Normalform**: Die Tabelle befindet sich in der 3. NF und keine Teile der Schlüsselkandidaten sind von Nicht-Schlüsselattributen funktional abhängig.