3 Grundlagen von Datenbanksystemen

Um die im letzten Kapitel genannten Bibliotheksfunktionen automatisieren zu können, wird ein EDV-System, das die Anforderungen erfüllen kann, eingesetzt. Da in einer Bibliothek große Datenmengen anfallen, sind die Anforderungen an das Zeitverhalten einer Datenverwaltung und nach Sicherheit der Daten gegenüber Benutzer- und Rechnerfehlern sehr hoch. Hinzu kommt, daß das EDV-System in der Lage sein muß, sehr flexibel mit den Daten umzugehen, da häufig Veränderungen und Neuerungen vorgenommen werden.

Basis eines EDV-Systems zur Automatisierung der Bibliotheksfunktionen bildet ein Datenbanksystem. Im folgenden wird die Entwicklungsgeschichte der Datenbanksysteme beschrieben. Anschließend wird die Architektur eines Datenbanksystems vorgestellt, die von der ANSI/SPARC-Gruppe im Jahre 1975 vorgeschlagen wurde. Im dritten Teil dieses Kapitels werden verschiedene Datenmodelle wie hierarchisches, netzwerkartiges, relationales Datenmodell und Entity/Relationship-Modell vorgestellt. Das relationale Datenmodell wird ausführlich behandelt, weil das zu spezifizierende Bibliothekssystem auf einem relationalen Datenbanksystem basieren soll. Am Ende des Kapitels werden die Eigenschaften von Programmier- und Datenbanksprachen und die Kopplungsarten zwischen den beiden Sprachen beschrieben.

3.1 Entwicklungsgeschichte der Datenbanksysteme

In diesem Abschnitt wird der Entwicklungsprozeß ausgehend von der Dateiverwaltung hin zu Datenbanksystemen skizziert.

3.1.1 Separate Dateiverwaltung

In den 50er und Anfang der 60er Jahre beherrschte vorwiegend eine verarbeitungsorientierte Sicht die Entwicklung von Anwendungssystemen.

Bei der separaten Dateiverwaltung werden Eingabedaten von Programmen sequentiell aus der Eingabedatei gelesen, bearbeitet und die Ausgabedaten in Ausgabedateien abgespeichert. Die nachfolgenden Programme verwenden die Ausgabedateien als Eingabedateien. Es gibt keine zentrale Verwaltung für Dateien.

Die Sicht des Anwendungsprogramms auf eine Datei entspricht der physikalischen Struktur der Daten. Die Datenstruktur wird für eine einzige Anwendung konstruiert und für sie optimiert.

3.1.2 Zentrale Dateiverwaltungssysteme

Mitte der 60er Jahre führten Weiterentwicklungen auf dem Gebiet der sekundären Speichermedien zur Einführung von Dateiverwaltungssystemen.

Es war möglich, Daten, die bislang auf mehreren Speichermedien verteilt waren, unter einer einheitlichen Schnittstelle zusammenzufassen. Wichtige Eigenschaften der als Teil des Betriebssystems realisierten Dateiverwaltungssysteme (wie z. B. ISAM) sind u. a. direkter Zugriff auf einzelne Datensätze, Datensicherheit und Online-Betrieb [4].

Durch die Möglichkeit, über Adressen direkt auf einzelne Datensätze mittels Index-Dateien oder Hash-Funktionen zuzugreifen, konnte der Mehrfach-Zugriff auf Dateien realisiert werden. Aber diese Realisierung des Mehrfachzugriffs stellt erhöhte Anforderungen an Zuverlässigkeit, Sicherheit, Integrität und Konsistenz des Datenbestandes. Diese sind jedoch von Dateiverwaltungssystemen nur schwer zu realisieren.

Diese Form der Datenverarbeitung hat folgende Schwachpunkte ([4], [6]):

- Die bereits vorhandenen Datenstrukturen zu erweitern bzw. zu verkleinern ist unmöglich, ohne die gesamte Datei neu zu erzeugen und alle Anwendungsprogramme, die diese Datei verwenden, zu ändern. Erfolgt eine Modifikation der Dateien, müssen im Regelfall alle darauf zugreifenden Programme angepaßt werden (*Logische Datenabhängigkeit*).
- Wenn das Speichermedium gewechselt wird oder wenn die Reihenfolge der Datenelemente in einem Datensatz geändert wird, dann müssen alle beteiligten Programme angepaßt werden, da die Anwenderprogramme direkt die Dateien verwalten (*Physische Datenabhängigkeit*).
- Daten müssen immer noch redundant gehalten werden. Dabei erweist sich die Sicherung der Konsistenz und Integrität des Datenbestandes als problematisch (*Datenredundanz*).
- Die Probleme mit redundanten Daten verursachen inkonsistente Datenbestände bei den Operationen (Einfügen, Löschen und Modifizieren). D. h. Dateien, die Daten zur selben Information führen, können unterschiedliche Einträge enthalten (*Dateninkonsistenz*).

3.1.3 Datenbanksysteme

Aus heutiger Sicht kann gesagt werden, daß für viele Anwendungen Dateiverwaltungssysteme nicht mehr ausreichen, weil die Anforderungen zu groß geworden sind. Für eine einfache Adressverwaltung sollten die Dateiverwaltungssysteme auch noch weiterhin genutzt werden, da der Umgang mit ihnen weniger aufwenig ist als mit Datenbanksystemen.

Im Vergleich zu Dateiverwaltungssystemen zeichnen sich Datenbanksysteme durch einen integrierten Datenbestand und weitgehende Unabhängigkeit zwischen Daten und Programmen aus. Der Begriff „Datenbanksystem“ bezeichnet ein rechnergestütztes System zur Beschreibung, Speicherung und Wiedergewinnung umfangreicher Datenmengen [4].

In diesem Abschnitt werden der Aufbau eines Datenbanksystems und die Grundbegriffe der Datenbanktechnik beschrieben.

3.1.3.1 Aufbau eines Datenbanksystems

Ein Datenbanksystem ist ein System zur Beschreibung, Speicherung und Wiedergewinnung von umfangreichen Datenmengen, die von mehreren Anwendungsprogrammen benutzt werden. Da die Realität sehr komplex ist, können die Daten nicht vollständig modelliert werden. Ein Abschnitt der realen Welt, der in einer Datenbank modelliert wird, nennt man *Diskursbereich*.

Die Daten, die aus dem Diskursbereich abstrahiert werden, werden in Form von Datenfeldern in der Datenbank gespeichert. Ein *Datenfeld* ist die kleinste Einheit der Daten, die für ihre Anwender von Bedeutung sind. Neben dieser Bezeichnung Datenfeld gibt es noch weitere wie *Datenelement* oder *Elementardatum*. Die Datenfelder werden zu Datensätzen zusammengefaßt. Ein Programm liest und schreibt normalerweise ganze Datensätze. Ein Satz von Datenfeldern kann in verschiedener Weise zu Datensätzen für die verschiedensten Zwecke zusammengesetzt werden [9]. Eine Datenbank besteht aus Datensätzen.

Grunsätzlich besteht ein Datenbanksystem aus der Datenbasis (Datenbank) und einem Programm zur Verwaltung der Daten in der Datenbasis (Datenbankverwaltungssystem).

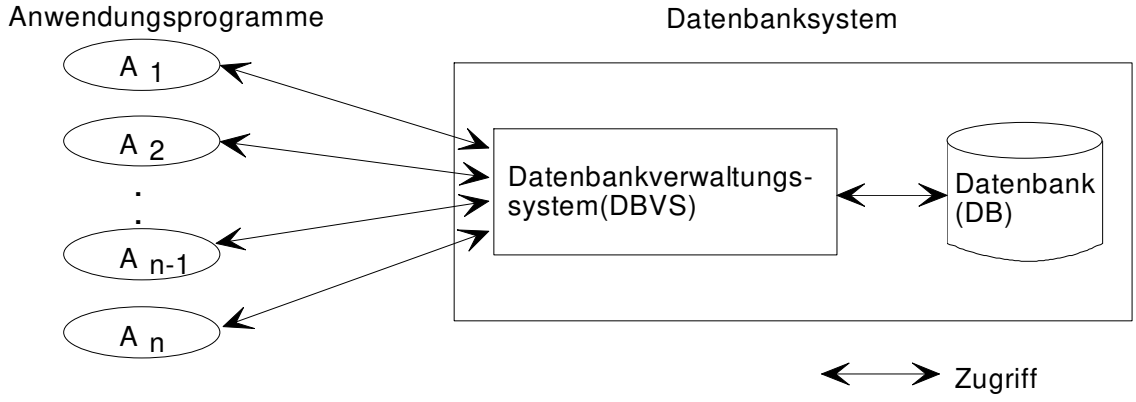

Abbildung 3.1.3.1.1 Aufbau eines Datenbanksystems

In Abbildung 3.1.3.1.1 wird der Aufbau eines Datenbanksystems gezeigt. Eine *Datenbank* ist eine integrierte, strukturierte Sammlung von Daten, die eine zentrale Speicherung und Kontrolle der Daten ermöglicht. Es können Beziehungen zwischen Dateien hergestellt werden.

Die zentrale Verwaltung und Kontrolle der Daten des Datenbestandes erfolgt durch das *Datenbankverwaltungssystem*, das außerdem den Datenzugriff der Anwendungsprogramme, sowie die Synchronisation der Zugriffe realisiert.

Zur Beschreibung, Abfrage und Verwaltung von Daten werden die *Datendefinitionssprache* (Data Definition Language: DDL) und die *Datenmanipulationssprache* (Data Manipulation Language: DML) benötigt. Die Datendefinitionssprache hat die Aufgaben, die logische Datenstruktur zu beschreiben, Wertebereiche für jedes Elment eines Datensatzes zu definieren, Zugriffsrechte zu bestimmen und eindeutige Felder zur Identifikation bestimmter Datensätze zu benennen. Die Datenmanipulationssprache dienen zur Formulierung von Anfragen, Eingaben, Änderungen und Löschungen der Datensätze in Datenbanken.

3.1.3.2 Datenbanksystem-Konzept

Die aus der Anwendung von Dateiverwaltungssystemen resultierende Problematik wird im Konzept der Datenbanksysteme durch die Integration aller Daten in einer Datenbank und durch die Schaffung einer zentralen Sicht auf den Datenbestand vermieden. Das Datenbankkonzept wird im folgenden beschrieben ([4], [5], [6], [9]):

- *Redundanzfreie bzw. -arme Datenhaltung*: Alle Informationen werden nur einmal abgespeichert, so daß keine Redundanz mehr vorliegt und Änderungen nur an einer Stelle

durchgeführt werden müssen. Werden Daten aus Effizienzgründen redundant gehalten, so ist durch ihre zentrale Kontrolle gewährleistet, daß an ihnen vorgenommene Änderungen nicht zu einem inkonsistenten Datenbestand führen.

- *Physische Datenunabhängigkeit*: Physische Datenunabhängigkeit bedeutet, daß die physikalische Darstellung (Speicherungsstruktur) und Organisation der Daten geändert werden kann, ohne die globale logische Struktur der Daten oder die Anwendungsprogramme ändern zu müssen. Daher muß der Anwender keinerlei Wissen über den physischen Aufbau der Daten besitzen, da er nur auf abstrakten Datenelementen arbeitet, die das System in physische Elemente umsetzt.
- *Logische Datenunabhängigkeit*: Logische Datenunabhängigkeit bedeutet, daß die globale logische Struktur der Daten geändert werden kann, ohne daß die Anwendungsprogramme geändert werden müssen. Deshalb können die Daten eines Anwendungsprogramms geändert werden, ohne andere Anwendungsprogramme zu beeinflussen.
- *Datenkonsistenz*: Unter Datenkonsistenz wird die logische Widerspruchsfreiheit einer Datenbank verstanden. Da die integrierte Datenhaltung zu einem weitgehend redundanzfreien Datenbestand führt, wird die Gefahr von Dateninkonsistenzen minimiert.
- *Datenintegrität*: Unter Datenintegrität wird die Korrektheit und Vollständigkeit einer Datenbank verstanden. Aufgrund der zentralen Kontrollmöglichkeit ist eine vereinfachte Überprüfung des Datenbestandes auf Korrektheit und Vollständigkeit möglich.
- *Sicherheit*: Es geht hier um die Verhinderung des Datenmißbrauchs bei der Verwendung der Daten.
- *Strukturflexibilität*: Die Datenbank ist nicht nur für die existierenden Anwendungen einsetzbar, sondern auch für neue Bedürfnisse der Benutzer. Es besteht die Möglichkeit, die bestehende Struktur zu ändern.
- *Antwortzeitverhalten*: Bei normaler Belastung ist das Antwortzeitverhalten akzeptabel.

3.2 Architektur von Datenbanksystemen: Drei-Ebenen-Architektur von ANSI/SPARC

Das zentrale Ziel einer Datenbank, die Integration von Daten und die Datenunabhängigkeit der Anwendungsprogramme, setzt voraus, daß sich alle Anwendungsprogramme auf eine einheitliche Beschreibung der Daten einigen. Um diese in der Praxis schwierige Entwurfsaufgabe zu erfüllen, wurde im Jahre 1975 die Drei-Ebenen-Architektur von der ANSI/SPARC-Gruppe vorgeschlagen.

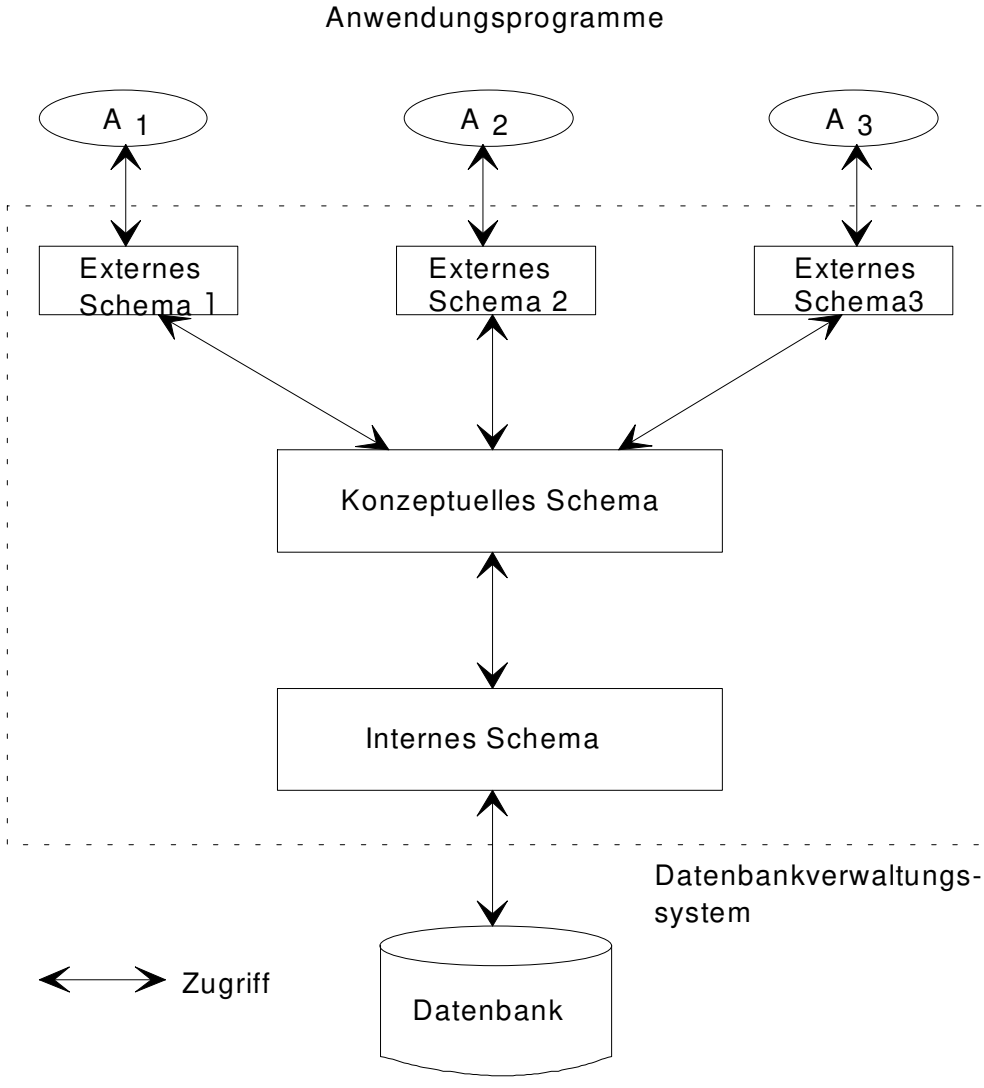

Abbildung 3.2.1 Drei-Ebenen-Architektur nach ANSI/SPARC

Wie in Abbildung 3.2.1 dargestellt, erfolgt die Beschreibung der Daten einer Datenbank nach diesem Konzept auf drei verschiedenen Ebenen, die jeweils durch eine andere Sichtweise geprägt sind. Zur Formulierung einer Sichtweise dienen Schemata. Ein Schema ist eine in einer Datenbeschreibungssprache abgefaßte Definiton der in einer Datenbank zugelassenen Datenstrukturen. Veränderungen der einzelnen Sichten können vorgenommen werden, ohne dadurch Einfluß auf die anderen Sichten zu nehmen.

Die Verbindung und die Kommunikation zwischen den einzelnen Ebenen respektive den unterschiedlichen Sichten erfolgt über Schnittstellen, die vom Datenbankverwaltungssystem bereitgestellt werden. Durch Transformationsregeln erfolgt die jeweilige Umsetzung zwischen internem und konzeptuellem oder konzeptuellem und externem Schema.

Durch die Einführung der Ebenen-Architektur können diese unabhängig voneinander dargestellt werden. Die Ebenen können im einzelnen wie folgt beschrieben werden ([4], [5]).

3.2.1 Externes Schema

Die Datenbank aus Sicht eines jeden Anwendungsprogramms wird auf der externen Ebene berücksichtigt und durch externe Schemata beschrieben. D.h. die externe Ebene repräsentiert die Datenbank aus der Sicht einzelner Anwendungen bzw. Benutzer. In den externen Schemata ist festzulegen, wie die Daten von den einzelnen Anwendungen oder Benutzern gesehen werden.

3.2.2 Konzeptuelles Schema

Auf der konzeptuellen Ebene erfolgt die logische Beschreibung der in einer Datenbank verwaltenden Daten. Durch Datenmodelle wird diese logische Gesamtsicht der Daten im konzeptuellen Schema beschrieben, d.h. unabhängig von Aspekten der physischen Datenorganisation oder Aspekten der Verwendung durch einzelne Benutzer. Auf diese Weise vermittelt das konzeptuelle Schema zwischen den externen Schemata der Benutzer und dem internen Schema. Mit dem konzeptuellen Schema wird folglich lediglich der Informationsgehalt der in einer Datenbank abgebildeten Daten erfaßt.

Während die externen Schemata die Sicht einer bestimmten Benutzergruppe bzw. eines Anwendungsprogrammes ausdrücken, beschreibt das konzeptuelle Schema die gesamte Datenbank. Das externe Schema muß natürlich verträglich mit dem konzeptuellen Schema sein.

3.2.3 Internes Schema

Auf die Ablage der Daten, ihre physische Abspeicherung, wird mit dem Begriff des internen Modells Bezug genommen. Mit ihm wird die Dateiorganisation, der Aufbau der physisch abgespeicherten Daten, die Zugriffsmöglichkeiten und die sonstige Organisation der Daten beschrieben.

3.3 Datenmodelle

Zur formalen Beschreibung aller in der Datenbank enthaltenen Daten und ihrer Beziehungen untereinander werden *Datenmodelle* auf der konzeptuellen und auf der externen Ebene verwendet.

Der Zweck eines Datenmodells läßt sich folgendermaßen formulieren [10]:

- Es soll eine anwendungsfachliche Vorstellung der Daten vermitteln, mit denen ein Informationssystem arbeitet. Es dient der Abstimmung mit dem Anwender und als Vorgabe für die Entwicklung.
- Es soll diese Daten detailliert und präzise definieren und strukturieren.
- Es soll die Grundlage für die Spezifikation der Funktionen des Informationssystems bilden.
- Es soll die Vorgabe für den technischen Entwurf der Datenbank sein.

Mit Hilfe eines Datenmodells werden die Datenstrukturen in einer abstrakten Weise beschrieben, wozu die Definition von Wertebereichen sowie die Modellierung von Beziehungen zwischen diesen Strukturen bzw. ihren Ausprägungen gehört.

Hinsichtlich der Art und Weise, wie die Datenmodelle Beziehungen zwischen den Objekten darstellen, sind das hierarchische Datenmodell, das Netzwerkdatenmodell und das relationale Datenmodell zu unterscheiden.

Die Abbildung von Informationsstrukturen der realen Welt auf das konzeptuelle Schema einer Datenbank wird meist nicht direkt vorgenommen, sondern vielmehr über den Zwischenschritt der Abbildung auf ein semantisches Datenmodell, das dann auf das konzeptuelle Schema transformiert wird. Das am häufigsten zu diesem Zweck eingesetzte Datenmodell ist das Entity/Relationship-Modell [6].

Zu den neueren Ansätzen gehören die objektorientierten , wissensbasierten, multimedialen Datenmodelle [5], auf die in dieser Arbeit nicht eingegangen wird.

3.3.1 Hierarchisches Datenmodell

Das hierarchische Datenmodell ist das älteste der Datenmodelle. Es wurde im Zusammenhang mit dem Datenbanksystem IMS (Information Management System) der Firma IBM Ende der 60er Jahre entwickelt.

Dieses Modell erlaubt nur baumförmige Beziehungen zwischen den Objekten, die bei der Erstellung der Datenbank fest angegeben werden müssen. Da es sich an den Maschinenanforderungen orientierte, wurde die Benutzerseite ungenügend berücksichtigt.

3.3.2 Netzwerkartiges Datenmodell

Das netzwerkartige Datenmodell wurde durch die CODASYL (Conference on Data Systems Languages) zwischen 1967 und 1971 entwickelt.

Die Strukturen im netzartigen Datenmodell bauen auf hierarchischen Bäumen auf. Die Knoten dienen der Darstellung von Objekten und die Kanten der Modellierung von Beziehungen. Im Gegensatz zum hierarchischen Modell sind die Kanten benannt und gerichtet [6].

3.3.3 Relationales Datenmodell

Das relationale Datenmodell zur Beschreibung von Datenbanken wurde 1970 von E. F. Codd vorgestellt [30]. Im Vergleich zu den oben genannten Datenmodellen besitzt das relationale Modell die einfachsten Datenstrukturen und die besten formalen Eigenschaften. Aus diesem Grunde ist es am weitesten verbreitet.

Von allen wichtigen Datenmodellen vollzieht das relationale am konsequentesten die Trennung zwischen logischer Datensicht und interner Realisierung. Auf der konzeptuellen Ebene werden keine Zugriffspfade definiert (wie es beim hierarchischen Modell und dem Netzwerkmodell der Fall ist). Stattdessen gibt es genau einen Datenstrukturtyp *Relation* (Tabelle). Verbindungen zwischen Tabellen können über gemeinsame Datenwerte in den Spalten hergestellt werden. Dies macht ein relationales Datenbanksystem offen für neue Anwendungen, die andere Verknüpfungen zwischen den Daten verlangen als diejenigen, die während der Konzeptionsphase bekannt waren [13].

Die logische Datenorganisation ist auch für den Datenbanklaien leicht durchschaubar - sogenannte Benutzersichten zeigen dem Anwender genau die Datenbankausschnitte, die er benötigt bzw. auf die er zugreifen darf. Komplexe Operationen können mit Hilfe komplexer Operatoren ausgeführt werden, die die Navigation übernehmen [13].

In den nächsten Kapiteln werden zuerst die Grundbegriffe bzw. -elemente des relationalen Datenmodells vorgestellt. Anschließend wird ein wichtiges Thema der Datenbanktheorie, die Redundanzvermeidung, in Form von Normalisierungsschritte behandelt. Als letztes werden die Operationen, die zur Datenverwaltung benötigt werden, beschrieben.

3.3.3.1 Grundelemente des relationalen Datenmodells

Alle Daten sowie Beziehungen zwischen Daten werden in zweidimensionalen Tabellen dargestellt. Im folgenden werden anhand eines Beispiels die Grundbegriffe des relationalen Datenmodells beschrieben.

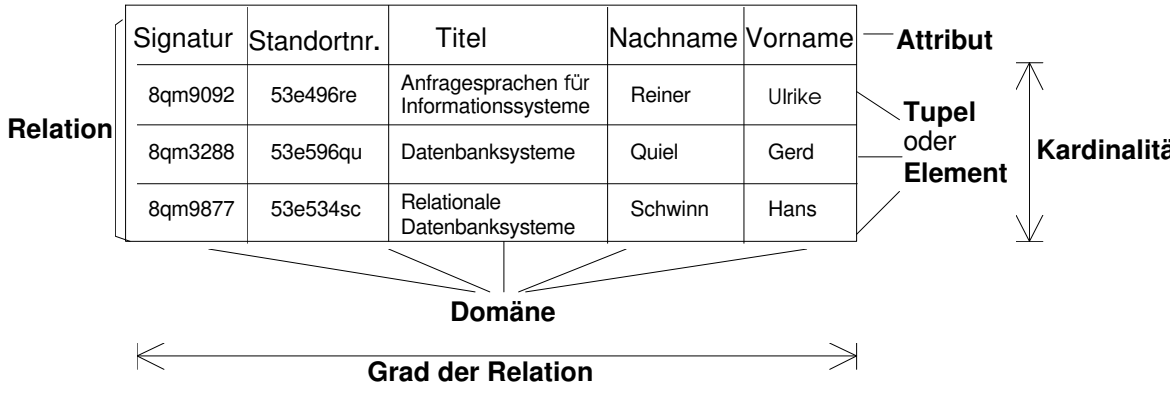

Abbildung 3.3.3.1.1 Beschreibung einer Relation durch eine Tabelle

Eine Tabelle wie in Abbildung 3.3.3.1.1 wird als *Relation* bezeichnet. Eine Zeile der Tabelle wird als *Tupel* bezeichnet. Zu einem Tupel sind die auf ein Objekt bezogenen Datenfeldwerte zusammengefaßt. Eine Spalte wird als *Attribut* bezeichnet, was einem Datenfeldtyp entspricht. Die Menge der verschiedenen Feldwerte (Ausprägungen) eines Attributes wird als *Domäne* bezeichnet [6]. Domäne bezeichnet die Konsistenzbedingungen, die für bestimmte Attribute gelten. Diese Bedingungen gehen über die in Programmiersprachen bekannten Datentypdeklarationen hinaus [4]. Die Anzahl der Tupel in einer Tabelle wird als *Kardinalität* bezeichnet. Die Anzahl der Attribute in einer Tabelle bestimmt deren *Grad*.

Daraus ergeben sich folgende Eigenschaften ([6], [9]):

- Jeder Eintrag in der Tabelle repräsentiert einen Datenfeldwert; es gibt keine Wiederholungsgruppen.
- Alle Einträge in einer Spalte sind homogen, d.h. alle Werte sind vom selben Typ.
- Jeder Spalte ist ein eindeutiger Name zugewiesen.
- Alle Zeilen sind voneinander verschieden, das doppelte Vorkommen gleicher Zeilen ist nicht erlaubt.
- Die Reihenfolge der Zeilen innerhalb der Tabelle ist beliebig.

Jedes Tupel muß einen *Schlüssel* besitzen, mit dem es identifiziert werden kann. Das Tupel kann durch ein einziges Attribut identifizierbar sein. Ein Primärschlüssel ist ein eindeutiger Schlüssel, mit dessen Hilfe normalerweise die Daten verarbeitet werden. (Z.B. *Signatur* in Abbildung 3.3.3.1.1) Oft ist jedoch mehr als ein Attribut notwendig, um ein Tupel in einer Relation zu identifizieren. *Schlüsselkandidat* ist ein Datenelement bzw. sind Datenelemente, die einen Schlüssel bilden können.

Zwischen den Relationen bestehen keine physischen Verweisstrukturen wie Indizes, Zeiger usw., sondern logische. Die logischen Verweise werden über *Fremdschlüssel* realisiert. Ein Attribut ist Fremdschlüssel, wenn seine Werte die Werte eines Schlüsselkandidaten einer anderen Relation sind [4].

Der Schlüssel muß zwei Eigenschaften haben [9]:

- Eindeutige Identifikation: Der Wert des Schlüssels in einem Tupel muß dieses eindeutig identifizieren.
- Redundanzfreiheit: kein Attribut aus dem Schlüssel kann weggelassen werden, ohne die Eigenschaft der eindeutigen Identifikation zu zerstören.

3.3.3.2 Anomalien und Normalformen

Um die redundante Datenhaltung und die auf Redundanz beruhenden Probleme wie die Inkonsistenz und Unübersichtlichkeit der Daten zu vermeiden oder möglichst zu vermindern, wird eine Normalisierung des relationalen Datenmodells durchgeführt. In diesem Abschnitt wird die Normalisierung bis zur dritten Normalform beschrieben[1] .

Eine Relation befindet sich in der *ersten Normalform* (1NF), wenn jeder Attributwert eine atomare, nicht weiter zerlegbare Dateneinheit ist, d.h. die Relation enthält keine Wiederholungsgruppe und ist daher eine flache Tabelle.

Bevor der nächste Normalisierungsschritt beschrieben wird, werden kurz die sogenannten drei Anomalien dargestellt, da diese das Motiv für die weiteren Normalisierungsschritte sind [5].

---

[1] Es gibt noch weitere Normalformen.

- Eine sogenannte *Update-Anomalie* liegt vor, wenn ein Wert an verschiedenen Stellen gespeichert wird und somit bei der Aktualisierung des Werts die Zahl der zu ändernden n-Tupel unbekannt ist.
- Eine *Einfüge-Anomalie* liegt vor, wenn beim Eintrag eines neuen Tupels einige Felder aus inhaltlichen Gründen leer gelassen werden müssen, was bei Schlüsseln nicht erlaubt ist.
- Von einer *Lösche-Anomalie* wird gesprochen, wenn beim Löschen einer Information, die nur einen Teil des n-Tupels betrifft, auch die übrigen Attributwerte verloren gehen.

Durch die oben genannten Anomalien sind noch diverse Integritätsverletzungen möglich. Dies liegt daran, daß ein Nicht-Schlüsselattribut nur von einem Teil des Schlüssels funktional abhängig ist[2] . Eine Relation liegt in der *zweiten Normalform* (2NF) vor, wenn sie in der 1NF ist und jedes Nichtschlüsselattribut voll funktional abhängig vom Primärschlüssel ist.

Auch bei Relationen, die sich bereits in zweiter Normalform befinden, ist noch die Möglichkeit redundanter Informationen vorhanden. Obwohl die Abhängigkeit von Attributen zu Teilschlüssel aufgelöst wurde, blieb eine mögliche Abhängigkeit zwischen Nichtschlüsselattributen unberücksichtigt [6]. Diese muß durch die dritte Normalisierung aufgelöst werden. Eine Relation ist in *dritter Normalform* (3NF), wenn sie in 2NF ist und kein Nichtschlüsselattribut transitiv vom Schlüsselattribut der Relation abhängig ist.
Es mag eingewandt werden, daß die Redundanz durch Normalisierung erhöht wurde. Normalisierung erfordert es, daß manche Datenfelder in mehr als einem Satz erscheinen, um die Sätze identifizieren zu können. Dieses Duplizieren hat nicht notwendig einen Anstieg im Speicherbedarf zur Folge, da die Normalisierung sich stets auf die logische Struktur - die Benutzersicht der Daten - bezieht, und nicht auf die Art und Weise, wie die Daten physikalisch auf dem Speicher dargestellt sind.

3.3.3.3 Operationen auf relationalen Datenbanken

Sprachen des relationalen Modells können mengenorientiert mit mathematischen Operationen arbeiten, weil das relationale Datenmodell auf mathematischen Theorien (Relationenalgebra, Relationenkalkül) basiert. Dies erlaubt die Formulierung des gewünschten Ergebnisses über Mengenoperationen, ohne durch die Datenstrukturen navigieren zu müssen.

---

[2] Funktionale Abhängigkeit entsteht, wenn eine Tabelle zusammengesetzte (mehr als zwei) Schüssel hat.

Die Operationen der Relationenalgebra auf Relationen sind Projektion, Selektion und Verbund. Sie werden im folgenden beschrieben:

- *Projektion*: Die Projektion wählt eine bestimmte Spalte einer Relation aus.
- *Selektion*: Die Selektion liefert diejenigen Zeilen einer Relation, die einer bestimmten Bedingung genügen. Diese Bedingung wird als Boolescher Ausdruck (z. B. mit den Operatoren UND, ODER, NICHT,...) formuliert.
- *Verbund (Join):* Die Verbund-Operation verknüpft zwei Relationen über je ein Attribut (Spalte). Diese beiden müssen einen gemeinsamen Wertebereich (Domäne) haben.

Jede Operation der Relationenalgebra auf Relationen erzeugt wieder eine Relation. Dadurch sind stets weitere Relationen-Operationen anwendbar; auch diese erzeugen wieder Relationen. Diese Eigenschaft relationaler Datenbanken ermöglicht die Einheitlichkeit der Datenmanipulation.

Da diese mathematischen Operationen jedoch für den ungeübten Benutzer unhandlich sind, gibt es eine weit verbreitete Datenbanksprache, *Structured Query Language* (SQL), die die Beschreibung der Anfrage in leicht verständlichen englischen Befehlen ermöglicht. Intern erfolgt die Umsetzung der Anfragen in mathematische Operationen [6].

3.3.3.4 Kritikpunkte am Relationenmodell

Häufig wird der geringe semantische Gehalt dieses Datenmodells kritisiert. Unter Semantik versteht man Informationen über die inhaltliche Bedeutung und Anwendung der Daten. Je mehr Semantik ein Datenmodell enthält, um so geringer ist der zusätzliche Aufwand in Anfragen oder Anwendungsprogrammen zur Auswertung der Daten [4].

Ein weiterer Kritikpunkt am relationalen Datenmodell ist die Schwierigkeit der Darstellung komplexer Datenobjekte, wie sie in Nonstandard-Datenbanken vorkommen, z. B. bei CAD-, Graphik- oder Computer-Animation-Anwendungen [4].

3.3.4 Entity/Relationship-Modell

Das Entity/Relationship-Modell (E/R-Modell) ist eine Entwurfsmethode, die in den Jahren 1975 und 1979 von Chen entwickelt und beschrieben wurde. Im Vergleich zu anderen Datenmodellen liegt das E/R-Modell auf einer semantisch höheren Ebene. Deshalb wird es

immer häufiger als Zwischenebene zwischen Benutzer und eigentlichen Datenmodellen (hierarchisch, netzartig, relational) genutzt.

Im folgenden werden kurz die Eigenschaften eines E/R-Modells beschrieben [4]:

- Das E/R-Modell ermöglicht die realitätsnahe und benutzerfreundliche Modellierung von Daten (semantisches Datenmodell) durch eine Trennung von Objekten und Beziehungen, die beim relationalen Modell verlorengeht.
- Es weist eine eingängige grafische Darstellungsmethodik auf und hat sich gerade im betrieblichen Umfeld für die Datenmodellierung bewährt. Zur Abbildung auf andere Datenmodelle existieren Möglichkeiten der Transformation, die eine systematische, weitgehend verlustfreie Abbildung erlauben.
- Die Transformation der gewonnenen Ergebnisse in ein ausführbares Datenbankschema ist weitgehend algorithmisierbar.
- E/R-Diagramme befinden sich in der dritten Normalform.

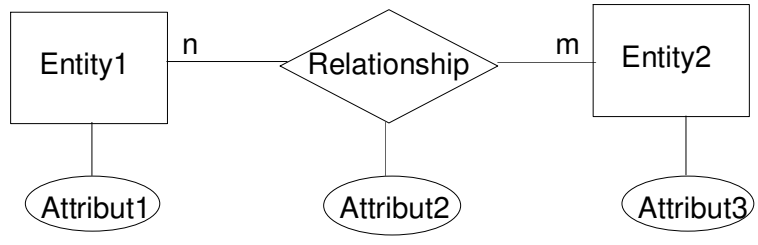

Abbildung 3.3.4.1 Darstellung eines E/R-Diagramms

Ein *Entity* ist ein individuelles und identifizierbares Objekt. Ein Entitytyp, das in Abbildung 3.3.4.1 als ein Rechteck gezeichnet ist, faßt eine Menge von Entities zusammen, die durch gemeinsame gleiche Eigenschaften gekennzeichnet sind ([14], [15]).
Beziehungen zwischen Entities werden durch sogenannte *Relationships* beschrieben, die ebenso wie Entities Datenobjekte sind. Eine Relationship muß wenigstens zwei Entities zueinander in Beziehung setzen. Im Gegensatz zu Entities ist eine Relationship jedoch kein eigenständiges Objekt, sondern kann nur existieren, wenn auch die referenzierten Entities existieren (referenzielle Integrität).

Ein Entity sowie eine Relationship wird durch ein oder mehrere *Attribute* in seinen Eigenschaften beschrieben. Einem Attribut ist ein Wertebereich (Domäne) zugeordnet, der die Menge der möglichen Attributwerte eines Entity oder einer Relationship definiert.

Zur näheren Spezifikation der Zahl von Entities, die durch Relationships zueinander in Beziehung gesetzt werden dürfen, können sogenannte *Kardinalitäten* für Relationship-Typen festgelegt werden.

3.4 Anwendungsprogrammierung

In diesem Abschnitt werden die Eigenschaften von Programmiersprachen und Datenbanksprachen beschrieben ([16], [27]). Die notwendige Kopplung zwischen Programmiersprachen und Datenbanksprachen wird dargestellt. Anschließend werden die Kopplungsarten Prozedurale Schnittstelle, einfache Sprachenerweiterung, Einbettung mit Vorübersetzer und Vierte-Generations-Sprache (4GL) vorgestellt. Als letztes wird 4GL anhand der Beipiele von INFORMIX-4GL näher erläutert ([4], [16]).

3.4.1 Eigenschaften von Programmier- und Datenbanksprachen

Programmiersprachen wie COBOL, FORTRAN, PASCAL, MODULA folgen dem operationalen Ansatz, der durch die von-Neumann-Maschine begründet wurde. Die Programmiersprachen dienen der Formulierung von imperativen Algorithmen und enthalten deshalb Elemente wie Kontrollstrukturen, Prozeduren, Module, Wertzuweisung und Typkonstruktion. Die Reihenfolge der Bearbeitung bei Programmiersprachen wurde vor der Ausführung festgelegt. Eine Programmiersprache ist demnach *berechnungsuniversell*, *satzorientiert* und *prozedural*.

Im Gegensatz zu den Programmiersprachen hat eine Datenbanksprache *deskriptiven* Charakter. Zur Formulierung von deskriptiven Anweisungen können ganze Objektmengen qualifiziert und wiedergefunden werden. Diese Anfragen beziehen sich immer auf ein fest vorgegebenes Datenmodell und werden meist nicht operational formuliert. Datenbanksprachen sind also *nicht berechnungsuniversell*, *mengenorientiert* und *deskriptiv*.

3.4.2 Kopplungsarten von Programmier- und Datenbanksprachen

Um ein Anwendungsprogramm für ein Datenbanksystem zu implementieren, werden Eigenschaften von Programmier- und Datenbanksprachen benötigt, da das Anwendungsprogramm Algorithmen (berechnungsuniversell) enthält und gleichzeitig die Daten in der

Datenbank manipuliert werden müssen. Aus dieser Notwendigkeit heraus wurde eine Kopplung von Programmier- und Datenbanksprachen versucht. Als nächstes werden vier Kopplungsarten erläutert.

1. *Prozedurale Schnittstellen* zeichnen sich dadurch aus, daß das jeweilige Anwendungsprogramm in einer frei wählbaren Programmiersprache geschrieben wird. Die Datenbankfunktionen werden über externe Prozeduren aufgerufen (z. B. hierarchisches Datenbanksystem IMS)
2. *Einfache Sprachenerweiterung*: Eine Programmiersprache wird um Sprachelemente einer Datenmanipulationssprache erweitert. Außerdem werden Programmiersprachenvariablen zum Informations- und Datenaustausch generiert (z. B. Programmiersprache COBOL wird um die entsprechenden CODASYL-Anweisungen erweitert).
3. Bei der *Einbettung mit Vorübersetzer* kommt man ohne Modifikation des Sprachübersetzers aus. Hier werden die Datenoperationen und -typen in einem Anweisungsprogramm durch ein besonderes Präfixsymbol (z. B. EXEC SQL) gekennzeichnet. (z. B. embedded SQL des Datenbanksystems DB2)
4. *4GL-Systeme*: 4GL ist eine Abkürzung für „Fourth Generation Language“. Meist versteht man unter einer Sprache der 4. Generation eine Mischung aus einer höheren Programmiersprache und einer (häufig relationalen) Datenbanksprache, dazu kommen oft noch Elemente einer „Job Control Language“ (z. B. INGRES-4GL, INFORMIX-4GL).

3.4.3 Anwendungsprogrammiersprache 4GL

In der Anwendungsprogrammiersprache 4GL werden deskriptive und prozedurale Sprachelemente integriert. Der Programmierer braucht sich weniger um die Ablauflogik zu kümmern, da die Sprachen mengenorientiert Daten selektieren. Sie enthält jedoch auch prozedurale Elemente, da nicht alle Probleme mit vordefinierten Prozeduren als Umsetzungen mengenorientierter Beschreibungen gelöst werden können.
Hauptbestandteile einer 4GL sind :

- Ein vorzugsweise relationales Datenbanksystem ist die Basis aller zu verwaltenden Daten.
- In einem erweiterten Datenkatalog werden die bekannten Systemrelationen gehalten, die benutzerdefinierte Relationen, Attribute, Sichten usw. beschreiben.
- Ein *Maskensystem* dient dem interaktiven Generieren von Eingabe- und Ausgabemasken für die Benutzungsschnittstellen der verschiedenen Anwendungen.

- Über ein Laufzeitsystem werden Bildschirmmasken, Ein-/Ausgabe und Überprüfungsroutinen aufgerufen.
- Ein *Report-Generator* ermöglicht die zweidimensionale Auswertung von Teilen der Datenbasis.
- Das Anfrage-Subsystem bietet dem Benutzer eine Anfragesprache des Basis-Datenbanksystems an.
- Die Anwendungssteuerung ist der integrierende Überbau über die restlichen Werkzeuge zur Generierung von vollständigen Applikationen. Die zugehörige Sprache wird häufig als die eigentliche 4GL-Sprache eines Gesamtsystems betrachtet.
- Als Erweiterungsfunktionen werden zusätzliche Werkzeuge bezeichnet. Z. B. Graphikbibliotheken zur Generierung von Balken- und Tortendiagrammen etc.. Oder mathematisch-statistische Funktionen, die der Report-Generator nutzen kann. Sowie Textverarbeitungsfunktionen, um komfortabel Serienbriefe, Angebote usw. erstellen zu können, und eventuell ein integriertes Tabellenkalkulationsprogramm, das insbesondere mit der Textverarbeitung und dem Report-Generator zusammenarbeitet.
- Alle diese Werkzeuge und Komponenten werden von vielen 4GL-Systemen unter einer einheitlichen, vielfach menügesteuerten Benutzungsschnittstelle zusammengefaßt.