# Relationale Datenbank

aus Wikipedia, der freien Enzyklopädie

Eine **relationale Datenbank** dient zur elektronischen Datenverwaltung in Computersystemen und beruht auf einem tabellenbasierten relationalen Datenbankmodell. Dieses wurde 1970 von Edgar F. Codd erstmals vorgeschlagen und ist bis heute trotz einiger Kritikpunkte ein etablierter Standard für Datenbanken.

Das zugehörige Datenbankmanagementsystem wird als **relationales Datenbankmanagementsystem** oder **RDBMS** (Relational Database Management System) bezeichnet. Zum Abfragen und Manipulieren der Daten wird überwiegend die Datenbanksprache SQL (Structured Query Language) eingesetzt.

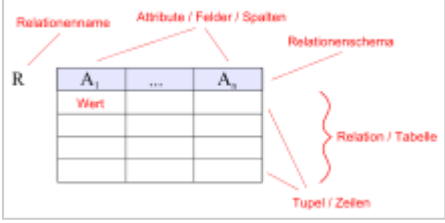
Grundlage des Konzeptes relationaler Datenbanken ist die Relation, ein im mathematischen Sinn wohldefinierter Begriff. Sie stellt eine mathematische Beschreibung einer Tabelle dar; siehe Datenbankrelation. Operationen auf diesen Relationen werden durch die relationale Algebra bestimmt. Die relationale Algebra ist somit die theoretische Grundlage von SQL.

## Inhaltsverzeichnis



## Grundlegende Konzepte

Eine relationale Datenbank kann man sich als eine Sammlung von Tabellen (den Relationen) vorstellen, in welchen Datensätze abgespeichert sind. Jede Zeile (Tupel) in einer Tabelle ist ein Datensatz (*record*). Jedes Tupel besteht aus einer Reihe von Attributwerten (Attribute = Eigenschaften), den Spalten der Tabelle. Das Relationenschema legt dabei die Anzahl und den Typ der Attribute für eine Relation fest. Das Bild illustriert die Relation $R$ mit Attributen $A_1$ bis $A_n$ in den Spalten.



Begriffe relationaler Datenbanken

Zum Beispiel wird ein Buch in einer Bibliothek durch den Datensatz *(Buch-ID, Autor, Verlag, Verlagsjahr, Titel, Datum der Aufnahme)* beschrieben. Ein Datensatz muss eindeutig identifizierbar sein. Das geschieht über einen oder mehrere Schlüssel (englisch *key*). In diesem Fall enthält *Buch-ID* die Schlüssel. Ein Schlüssel darf sich niemals ändern. Er bezieht sich auf den Datensatz und nicht auf die Position in der Tabelle.

**Beispiel einer Relation „Buch“:**

| Buch-ID | Autor | Verlag | Verlagsjahr | Titel | Datum |
|---|---|---|---|---|---|
| 1 | Hans Vielschreiber | Musterverlag | 2007 | Wir lernen SQL | 13.01.2007 |
| 2 | J. Gutenberg | Gutenberg und Co. | 1452 | Drucken leicht gemacht | 01.01.1452 |
| 3 | G. I. Caesar | Handschriftverlag | −44 | Mein Leben mit Asterix | 16.03.−44 |
| 5 | Galileo Galilei | Inquisition International | 1640 | Eppur si muove | 1641 |
| 6 | Charles Darwin | Vatikan-Verlag | 1860 | Adam und Eva | 1862 |

# Beziehungen zwischen Tabellen

Weiterhin können Verknüpfungen genutzt werden, um die Beziehungen zwischen Tabellen auszudrücken. Eine Bibliothekdatenbank könnte damit etwa folgendermaßen mit drei Tabellen implementiert werden:

Tabelle *Buch*, die für jedes Buch eine Zeile enthält:

- Jede Zeile besteht aus den Spalten der Tabelle (Attributen): *Buch-ID, Autor, Verlag, Verlagsjahr, Titel, Datum der Aufnahme*.
- Als Schlüssel dient die *Buch-ID*, da sie jedes Buch eindeutig identifiziert.

Tabelle *Nutzer*, die die Daten von allen registrierten Bibliotheksnutzern enthält:

- Die Attribute wären zum Beispiel: Nutzer-ID, Vorname, Nachname.

**Relation „Nutzer“**

| Nutzer-ID | Vorname | Nachname |
|---|---|---|
| 10 | Hans | Vielschreiber |
| 11 | Jens | Mittelleser |
| 12 | Erich | Wenigleser |

**Relation „Entliehen“**

| Nutzer-ID | Buch-ID |
|---|---|
| 10 | 1 |
| 10 | 2 |
| 10 | 3 |
| 12 | 5 |
| 12 | 6 |

Außerdem braucht man eine dritte Tabelle *Entliehen*, die Informationen über die Verfügbarkeit des Buches enthält. Sie würde die Attribute *Nutzer-ID* und *Buch-ID* enthalten. Jede Zeile dieser Tabelle *Entliehen* ordnet einer Nutzer-ID eine Buch-ID zu.

Der Eintrag *(10,3)* würde also heißen, dass der Nutzer mit der ID *10* („Hans Vielschreiber“) das Buch mit der ID *3* („Mein Leben mit Asterix“) entliehen hat. Derselbe Nutzer hat auch das Buch „Drucken leicht gemacht“ entliehen, was durch den Tabelleneintrag *(10,2)* belegt ist. Als Schlüssel nimmt man hier die Attributmenge *(Nutzer-ID, Buch-ID)*. Gleichzeitig verbindet die *Nutzer-ID* jeden Eintrag der Tabelle *Entliehen* mit einem

Eintrag der Tabelle *Nutzer*, sowie die Buch-ID jeden Eintrag von *Entliehen* mit einem Eintrag der Tabelle *Buch* verbindet. Deswegen heißen diese Attribute in diesem Zusammenhang Fremdschlüssel (engl. *foreign key*). Tabellen ohne Fremdschlüssel nennt man flache Tabellen.

Der hier benutzte Begriff *Relation* beschreibt nicht die Beziehung zwischen Entitäten (wie im Entity-Relationship-Modell), sondern die Beziehung der Attribute zum Relationennamen. So gilt im obigen Beispiel: *Hans ist Vorname (Attribut) von Nutzer (Relationenname)*. Außerdem wird *Relation* bei relationalen Datenbanken allgemein als Synonym für *Tabelle* genutzt (meist aus *Entitätstyp* im ERM entstehend).

# Abgrenzung

Neben dem relationalen Datenbankmodell gibt es verschiedene alternative Konzepte, die es erlauben, Daten in anderen Strukturen zu verwalten. Diese Konzepte haben oft nur noch eine geringe Bedeutung oder haben sich noch nicht durchgesetzt. Dennoch bieten sie für bestimmte Applikationen eine einfachere Anbindung der zu verwaltenden Daten.

## Ältere Ansätze

In den 60er und 70er Jahren wurden zur betrieblichen Datenverarbeitung hierarchische Datenbanksysteme sowie Netzwerk-Datenbanksysteme verwendet. Bei diesen wird die Daten- bzw. Tabellenstruktur in der Entwurfsphase definiert und kann nicht bei der Abfrage variiert werden. Sie kommen in Spezialfällen auch heute noch zum Einsatz.

## Objektorientierte Datenbanken

Mit dem Aufkommen objektorientierter Programmiersprachen wurden zunehmend Objektdatenbanken angeboten. Damit können Objekte aus OO-Sprachen wie Java direkt in der Datenbank gehalten werden – eine Abbildung der Objekte auf die relationale Tabellenstruktur, das objektrelationale Mapping, ist dann nicht mehr notwendig. Dieses Vorgehen hat Vorteile gegenüber dem relationalen Entwurf, wenn man komplexe Datenobjekte speichern möchte, die nur schwer auf die flachen relationalen Tabellenstrukturen abgebildet werden können. Objektdatenbanken haben jedoch noch immer Nachteile gegenüber relationalen Datenbanken bei der Verarbeitung großer Datenmengen. Dies ist beispielsweise durch Zugriffspfade zu Objekten über mehrere Pfadarten (bspw. Vererbung und Assoziation) verursacht. Dies führt bei Schreiboperationen in der Sperrverwaltung zu einer exponentiellen Komplexität und somit zu schlechter Leistung. Die Leistungsprobleme wurden in den objektrelationalen Datenbanken aufgegriffen, in denen nur die Konstrukte aus objektorientierten Datenbanken mit niedrigerer Komplexität (bspw. $O(n \cdot \log(n))$) übernommen wurden.

## Objektrelationale Datenbanken

Einige Anbieter fügen ihren relationalen Datenbanken objektorientierte Eigenschaften hinzu und nennen diese dann objektrelationale Datenbanken. Diese sind jedoch nicht zur direkten Abbildung von Objekten der Programmiersprache vorgesehen – sie benutzen lediglich das Konzept der Vererbung bei Definition und Abfrage von Tabellen mit ähnlichen Feldstrukturen und vereinfachen damit deren Handhabung. Der SQL-99-Standard wurde um objektrelationale Sprachelemente erweitert.

## Semistrukturierte Datenbanken

Neuere Konzepte sind die semistrukturierten Datenbanken. Sie unterscheiden sich von den herkömmlichen Datenbankmodellen darin, dass sie kein fest vorgegebenes Schema haben. Die Datenbank wird hierarchisch, baumartig aufgebaut und jede Datenbankeinheit (engl. *entity*) des gleichen Typs kann verschiedene Mengen von *Attributen* haben.

Typische Vertreter dieses Typs sind XML-Datenbanken, welche die Daten als XML-Fragmente verwalten. Die XML-Daten sind hierbei hierarchisch geordnet und können beliebige Strukturen enthalten, solange diese nach XML-Definition wohlgeformt sind. Die Daten können über XQuery oder XPath abgefragt werden. Zur Manipulation werden heute proprietäre Spracherweiterungen verwendet. Nachteil von aktuellen XML-Datenbanken ist die im Vergleich zu relationalen Systemen geringere Performance.

Semistrukturierte Datenbanken lassen sich über Erweiterungen oder Server-Programmierung auch mit relationalen DB realisieren, wobei das Relationenmodell aber nicht mehr angewendet wird.

# Theorie der relationalen Datenbanken

Die Grundlagen der Theorie der relationalen Datenbank wurden von Edgar F. Codd in den 1960ern und 1970ern gelegt und in seiner Arbeit *A Relational Model of Data for Large Shared Data Banks*[1] beschrieben. Theoretisch basieren alle Operationen auf der relationalen Algebra.

## Grundlegendes zur relationalen Algebra

Die relationale Algebra ist ein algebraisches Modell, das beschreibt, wie Daten gespeichert, abgefragt und manipuliert werden können. Die wesentlichen Operationen, aus denen alle weiteren abgeleitet werden können, sind die folgenden:

- Projektion (engl. *projection*)
- Selektion (engl. *selection*)
- Kreuzprodukt oder Kartesisches Produkt (engl. *cross product*, *cross join* oder *cartesian product*)
- Umbenennung (engl. *rename*)
- Vereinigung (engl. *union*)
- Differenz (engl. *set difference* oder *minus*)

Alle Anfragen, die mittels SQL an eine relationale Datenbank gestellt werden, werden vom Datenbankmanagementsystem auf diese Operatoren abgebildet, das heißt übersetzt. In der Praxis gibt es weitere Operatoren, wie zum Beispiel den Join-Operator, der jedoch ebenfalls nur eine Kombination aus Kreuzprodukt, Selektion und Projektion darstellt.

## Beschränkungen der relationalen Algebra

Die relationale Algebra bietet keine Unterstützung zur Berechnung von rekursiven Anfragen (transitive Hülle). Das heißt beispielsweise, dass es nicht möglich ist, in einer Anfrage alle Vorfahren einer Person zu berechnen, wenn diese in einer Relation *Person* gespeichert sind und über eine Relation *VorfahreVon* mit dem jeweiligen Vorfahren in *Person* verbunden ist. Die Vorfahren können nur durch eine Folge von Anfragen ermittelt werden.

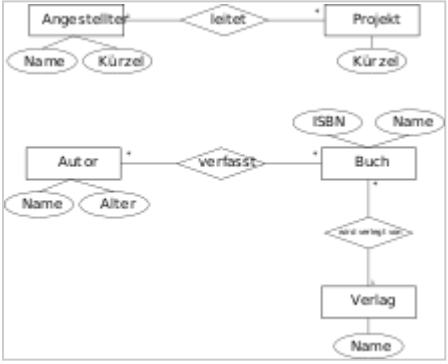
Mit der Einführung von SQL-99 wurde jedoch auch eine erweiterte relationale Algebra eingeführt, die eine Operation zur Berechnung der transitiven Hülle erlaubt.

# Datenbankschema und Modellierung

Wichtiger Bestandteil einer relationalen Datenbank ist ihr *Schema*. Das Schema legt fest, welche Daten in der Datenbank gespeichert werden und wie diese Daten in Beziehung zueinander stehen. Der Vorgang zum Erstellen eines Schemas nennt sich Datenmodellierung.

Zur Modellierung von relationalen Datenbanken wird auch das Entity-Relationship-Modell verwendet. Es dient zum Entwurf eines konzeptuellen Schemas, welches unter Verwendung eines Datenbankmanagementsystems (DBMS) implementiert werden kann. Dieser Schritt wird als *logischer Entwurf* oder auch *Datenmodellabbildung* bezeichnet und hat als Ergebnis ein Datenbankschema im Implementierungsdatenmodell des DBMS.

Ein wichtiger Schritt des Modellierungsprozesses ist die Normalisierung. Diese soll Redundanzen verringern und Anomalien verhindern, um so die Wartung einer Datenbank zu vereinfachen, sowie die Konsistenz der Daten zu gewährleisten. Edgar F. Codd hat dazu vier Normalformen vorgeschlagen, die seitdem bei dem relationalen Datenbankentwurf zum Einsatz kommen und um weitere ergänzt wurden.



Ein ER-Diagramm in Chen-Notation

# Kritik am relationalen Datenbankmodell

**Segmentierung**
: In der relationalen Darstellung erfolgt die Abspeicherung eines Objektes segmentiert auf viele unterschiedliche Relationen. Die Anwendungsobjekte sind normalerweise *komplex*, bestehen also selbst wieder aus Objekten oder Listen von Objekten. Da das relationale Modell lediglich Tupelmengen kennt, die aus *Werten* bestehen, müssen komplexe Anwendungsobjekte bei einer Abfrage durch das DBMS mittels zahlreicher Joins aus den einzelnen Relationen wiederhergestellt werden. Dies kann zu unübersichtlichen Abfragen führen, die bei jeder strukturellen Änderung des Anwendungsobjekts auf Anpassungsbedarf hin überprüft werden müssen. Die Verwendung von Joins, welche durch jeweils gut passende Datenbank-Indizes unterstützt werden müssen, macht den Objektzugriff aufwendiger als z. B. bei einer Objektdatenbank, sowohl beim Ressourcenbedarf als auch beim Entwicklungsaufwand.

**Künstliche Schlüsselattribute**
: Zur eindeutigen Identifizierung von Tupeln müssen in manchen Fällen künstliche Schlüssel eingesetzt werden. Dies dient z. B. dazu, die Größe des Schlüssels zu reduzieren, wenn er als Fremdschlüssel eingesetzt werden soll, oder dazu, gehört-zu-Beziehungen zu implementieren. Es werden also Attribute in die Relation aufgenommen, die mit der abstrakten Beschreibung eines Anwendungsobjektes nichts zu tun haben, sondern „nur“ Verwaltungsinformationen sind.

**Externe Programmierschnittstelle**
: Da in vielen relationalen Datenbanken nur Datenmanipulationssprachen eingeschränkter Mächtigkeit vorhanden sind, werden meist Schnittstellen zu mächtigeren Programmiersprachen notwendig. Diese Verbindung führt ggf. zu einer ungünstigen Handhabung, z. B. wenn das mengenorientierte SQL in dem satzorientierten C++ zu verarbeiten ist, siehe Impedance Mismatch.
Es gibt jedoch auch relationale Datenbanken mit mächtigen Programmiersprachen, etwa PL/SQL in Oracle oder PL/pgSQL in PostgreSQL oder T/SQL in Microsoft SQL Server; bei beiden Datenbanksystemen erlauben die jeweiligen Datenmanipulationssprachen das Einbinden von anderen Programmiersprachen. PL/SQL ermöglicht z. B. die Verwendung von Java-Programmen oder C++-Programmen innerhalb von PL/SQL-Programmen. PL/pgSQL ermöglicht wiederum die Server-Programmierung mit anderen Sprachen wie PHP, Tcl oder Python.

**Objekteigenschaften und -verhalten häufig nicht abbildbar**
: In der relationalen Datenbank kann das anwendungstypische Verhalten eines Objektes nicht beschrieben werden. Diese Beschreibung kann somit erst außerhalb der Datenbank in einer Anwendungssoftware erfolgen. Wenn mehrere Anwendungen den gleichen Datenbestand nutzen, kann das zu einer redundanten Implementierung führen.

Mit dem Sammelbegriff NoSQL werden **nicht-relationale Datenbankmodelle** bezeichnet, die Probleme wie die genannten durch alternative Herangehensweisen zu lösen beabsichtigen.

# Gegenüberstellung von Grundbegriffen

| Relationale Datenbank | Relationen-Modell | Entity-Relationship-Modell (ERM) | Unified Modeling Language (UML) |
|---|---|---|---|
| Wertebereich (Domäne, Domain) | Wertebereich (Domäne, Domain) | Wertebereich (Domäne, Domain) | Wertebereich (Domäne, Domain) |
| Kopfzeile | Relationstyp/Relationsformat | Entitätstyp | Klasse, Objekttyp |
| Spalte | Attribut | Attribut | Attribut |
| Tabelle | Relation | Entitätsmenge(-set) | Objektmenge, Instanzmenge, Klasse |
| Spaltenüberschrift(en) | Fremdschlüssel | funktionale Beziehung (Relationship) | Assoziation |
| Zeile | Tupel | Entität | Objekt, Instanz |
| Zelle | Attributwert | Attributwert | Attributwert |

# Siehe auch

- Relationale Entwurfstheorie
- Datenbank-Muster
- Datenbankoperator
- Deduktive Datenbank
- Liste der Datenbankmanagementsysteme
- Referentielle Integrität
- Codd’sche Regeln
- SQL
- NoSQL

# Literatur


- Edgar F. Codd: *The Relational Model for Database Management. Version 2.* Addison-Wesley, Reading u. a. 1990, ISBN 0-201-14192-2.
- Alfons Kemper, André Eickler: *Datenbanksysteme. Eine Einführung.* R. Oldenbourg Verlag, München 2004, ISBN 3-486-27392-2.
- Andreas Meier: *Relationale und postrelationale Datenbanken*. Springer-Verlag, 7. Auflage, Heidelberg 2010, ISBN 978-3-642-05255-2 (online (http://www.springer.com/computer/database+management+%26+information+retrieval/book/978-3-642-05255-2)).


# Weblinks

- Kriterien für relationale Datenbanksysteme (http://v.hdm-stuttgart.de/~riekert/lehre/db-kelz/chap6.htm) mit ausführlicher Beschreibung der 12 Regeln
- *RDBMS*. (http://wikis.gm.fh-koeln.de/wiki_db/Datenbanken/RDBMS) In: Datenbanken Online Lexikon, FH Köln
- Datenbanken-verstehen (http://www.datenbanken-verstehen.de/) – Umfangreiches Portal
- Zur Zukunft von RDBMS: *Tunen statt ablösen*. (http://www.computerwoche.de/index.cfm?pid=746&pk=1235665#) Computerwoche, September 2012

# Einzelnachweise


1. Edgar F. Codd: *A Relational Model of Data for Large Shared Data Banks (http://www.acm.org/classics/nov95/)*. In: *Communications of the ACM*. ACM Press, 13. Juni 1970, ISSN 0001-0782 (http://dispatch.opac.d-nb.de/DB=1.1/CMD?ACT=SRCHA&IKT=8&TRM=0001-0782), S. 377–387.

Kategorie: Relationales Datenbankmanagementsystem

---