1. OKTOBER 2019

DB MODELLE

KOMPETENZNACHWEIS

NURIA ANAYA , ADRIANA ARTEAGA
ANDREAS NYDEGGER, KEVIN SCHLEUNIGER
TBZ – MODUL

Inhaltsverzeichnis

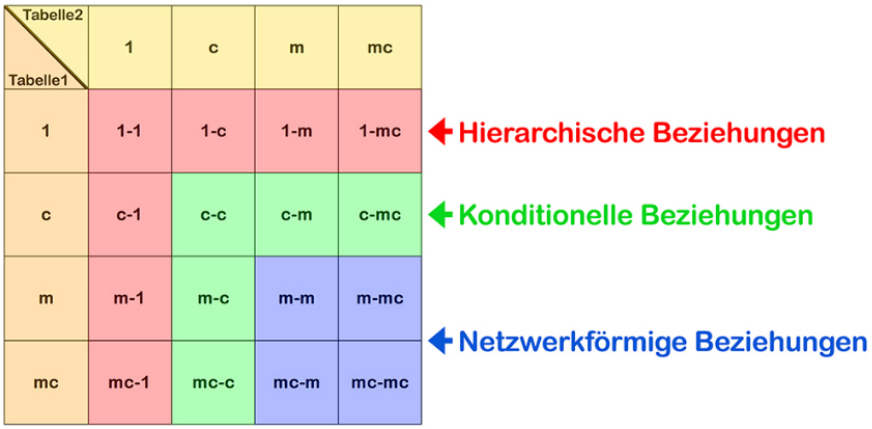
DB-Modelle

Ein Datenbankmodell bestimmt die Struktur die in einem Datenbanksystem gespeichert wird, d.h. die Tabellen, Beziehungen und die allg. Struktur.

Das Datenbankmodell definiert sich aus drei Eigenschaften:

1. Einer generischen Datenstruktur, die die Struktur einer Datenbank beschreibt
2. Generischen Operatoren
3. Integritätsbedingungen

Mögliche Beziehungstypen



Hierarchisches Modell

Das Hierarchische Modell ist das älteste Datenbankmodell und wird als hierarchische Baumstruktur dargestellt.
Zu den Eigenschaftes des Modells zählt das jeder Datensatz hat einen Vorgänger hat, mit Ausnahme des Datensatzes, der an der Wurzel der Baumstruktur ist (im unteren Beispiel der Kunde). Ebenfalls werden Verknüpfungen über Eltern-Kind-Beziehungen realisiert und dementsprechend in der Baumstruktur abgebildet.
Im Bereich der XML-Entwicklung hat das hierarchische Datenbankmodell in letzter Zeit wieder an Bedeutung gewonnen.

Durch die Baumstruktur lassen sich nur 1:1 und 1:n Beziehungen darstellen(siehe Beispiel Grafik unten). Die n:m Beziehungen können durch Redundanzen erreicht, besser aber über virtuelle Parent-Child-Relationship (VPCR)

Die minimalen Bedingungen an ein hierarchisches Datenbankmodel sind:

- Ein Record-Typ muss das Wurzelelement darstellen und ist somit kein „Child“
- Jeder andere Record-Typ tritt genau einmal als „Child“ auf
- Ein Record-Typ der nicht als „Parent“ auftritt, wird „Blatt“ genannt

Vorteil

- Einfach zu verstehen
- Einfacher Zugriff auf zusammengehörige Daten (z.B. Lieferant → Rechnung)
- Verarbeitung von Stücklisten möglich
- Eignet sich für XML-Daten

Nachteil

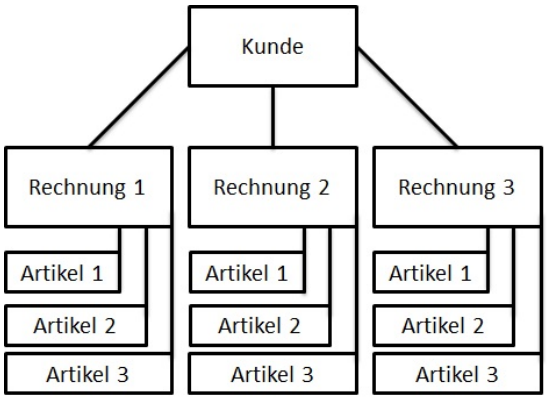
- Unflexibel, weil man immer über Hierarchie auf den gewünschten Satz kommt
- Man muss die Zugriffspfade kennen
- Keine n:m-Beziehung möglich
- Man kann nur eine Baumstruktur verwenden und kann nicht zwei Baumstrukturen miteinander verknüpfen

Anwendung

- Die bekannteste Anwendung des hierarchischen Datenbankmodells ist das Dateisystem (z.B. Explorer auf dem Computer)
- Oder für XML-Dateien

Beispiel

Beispiel für ein Hierarchisches Datenbankmodell



Netzwerkmodell

Das Netzwerkmodell ist auch unter dem Namen „CODASYL Datenbankmodell“ oder „DBTG Datenbankmodell“ bekannt. Es wurden drei Datenbanksprachen (DML, DDL, DCL) für dieses Modell entworfen und veröffentlicht.
Ein Datenfeld besteht aus einem Namen und einem Wert. Das Netzwerk-Modell fordert auch keine strenge Hierarchie und kann deswegen auch m:n-Beziehungen abbilden, d.h. ein Datensatz kann mehrere Vorgänger haben. Auch können mehrere Datensätze an oberster Stelle stehen. Es ist sozusagen eine Verallgemeinerung des hierarchischen Datenbankmodelles zu sehen. Jedoch wurde das Netzwerkdatenbankmodell niemals wirklich übernommen und schliesslich durch das relationale Modell ersetzt. Durch die direkte Verbindung der Knoten entsteht ein Netz, daher auch der Name.

Vorteil

- Unterschiedliche Suchwege als Lösungsweg, was auch zum Problem führen kann

Nachteil

- Unübersichtlichkeit beim ständigen weiter wachsen

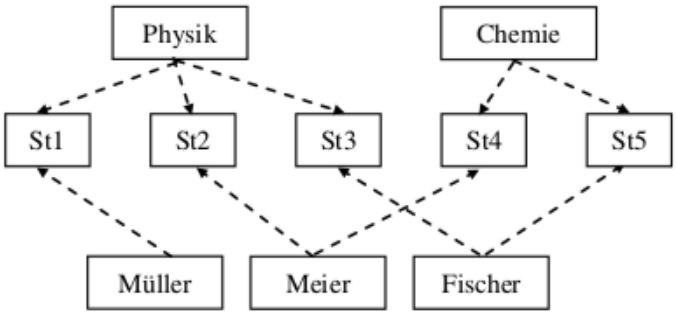
Anwendung

Eignen sich besonders gut

- zur Abbildung von netzartigen Strukturen z.B. geografische Orte und deren Verbindungen untereinander
- Internet und Intranet
- semantisches Web
- Personen- und Firmennetzwerke
- Grossrechner

Beispiel

Beispiel für ein Netzwerkdatenbankmodell



Relationales Modell

Das Relationale Datenbankmodell hat sich als effizientes und flexibles Datenbankmodell erweisen und ist heute das meist verwendete Modell.
Relationale Datenbanken basieren auf Relationen von Namen und zugehörigen Attributen. Die Relationen werden in Form zweidimensionaler Tabellen mit einer eindeutigen Bezeichnung. Jeder Datensatz ist eine Zeile in der Tabelle. Die Zeilen sind eindeutig über einen oder mehrere Primärschlüssel identifizierbar. Die Attributswerte bilden Spalten der Tabelle.

Vorteil

- Einfaches Datenmodell (einfache Implementierung & Verwaltung)
- Geringe Datenredundanz
- Hohe Datenkonsistenz
- Mengenorientierte Datenverarbeitung (ermöglicht Abfrage mit JOINS)
- Einheitliche Abfragesprache

Nachteil

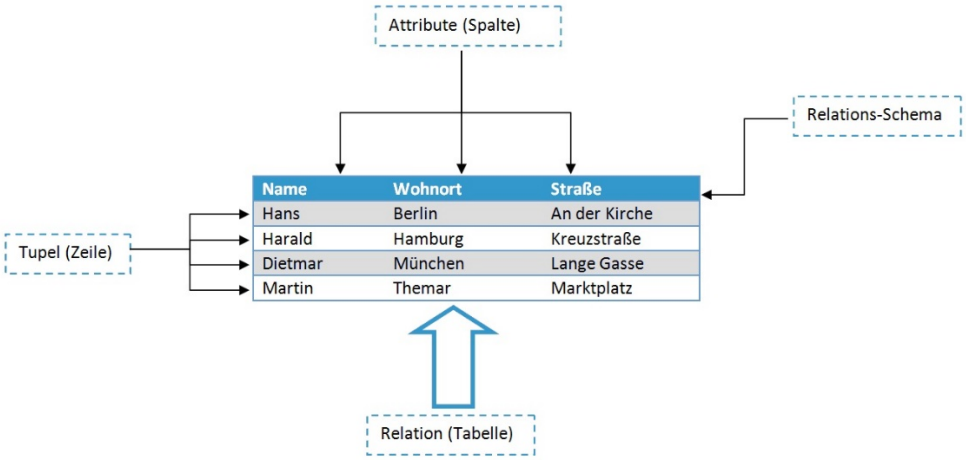
- Tabellarische Abbildung von Daten (nicht alle Datentypen lassen sich in ein starres Schema miteinander verknüpfen)
- Keine hierarchisches Datenbankschema
- Segmentierung von Daten (durch Normalisierung werden Daten aufgeteilt)
- Schlechte Performance in Vergleich zu NoSQL-Datenbanken

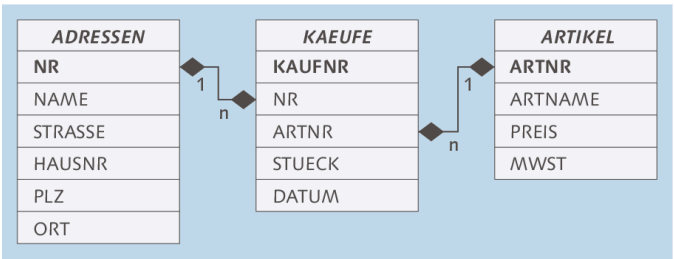
Anwendung

Das relationale Modell kann oft und fast überall verwendet werden.

Beispiel

Beschreibung einer relationalen Datenbanktabelle





Objektorientiertes Datenbankmodell

Das objektorientierte Datenbankmodell sieht die Speicherung von Daten als Objekte vor. Die Modulation von Objekten erfolgt dabei analog zur objektorientierten Programmierung. Ein Objekt definiert eine Entität und enthält:

- Die zur Beschreibung der Entität benötigten Eigenschaften(Attribute)
- Verweise (Beziehungen) zu andern Objekten sowie
- Funktionen, die den Zugriff auf die gespeicherten Daten ermöglichen (Methoden)

Die Abfrage der Daten kann über die Funktionen des Objektes oder mit der Objektabfragesprache erfolgen.

Vorteile

- Effizienter, wegen des nicht Trennens der Daten und der Datenbank
- Performance Steigerung dank objekteigene Funktionen für Abfrage oder Abfragesprache
- Speicherung multimedialer Inhalte, komplexer Datentypen

Nachteile

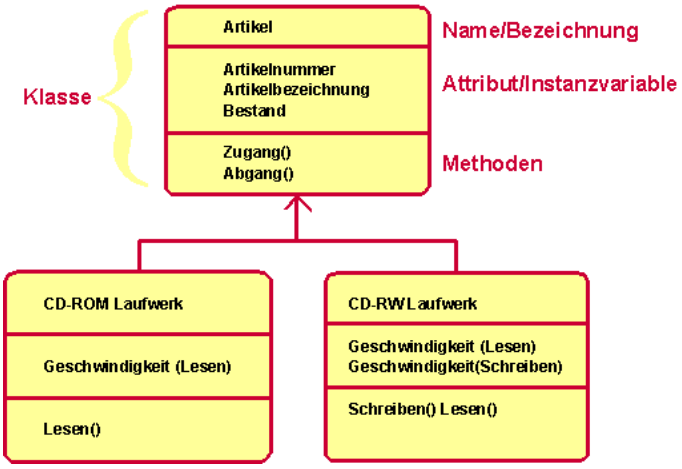
- Wenige kompatible Schnittstellen, wegen geringer Verbreitung

Anwendung

Wird bei Verwaltung komplexer Objekte angewendet.

Beispiel

Beispiel eines objektorientiertes Datenbankmodells



Objektrelationales Modell

Objektrelationales Datenbankmodell (auch **O**bjekt**r**elationales **D**aten**b**ank**m**anagement**s**ystem; kurz ORDBMS)unterstützen Datenobjekte und Operationen höherer Komplexität (z.B. Text- und Bilddaten, Audio usw.). Die Zugriffe auf die objektrelationale Datenbank werden durch Transaktionen gesteuert.
Bildet das Bindeglied zwischen relationalen Datenbanken und Objektdatenbanken.

Zwischen den relationalen und den objektorientierten Datenmodellen gibt es viele Entsprechungen, so entspricht die Entität dem Objekt und der Entitätstyp der Klasse. Diese Basis hat zur Entwicklung von objektrelationalen DBMS geführt, die durch eine Spracherweiterung um objektorientierte Methoden und Datentypen zur Verbesserung des relationalen Modells und damit zur Handhabung komplex strukturierter Daten geführt haben. Häufig wird über eine Relationale Datenbank eine objektorientierte Zugriffsschicht gesetzt. Bei manchen Zugriffsschichten werden die Objekte und deren Attribute erst dann geladen, wenn sie in der Anwendung auch benötigt werden.

Vorteile

- Handhabung komplexe Datenstrukturen

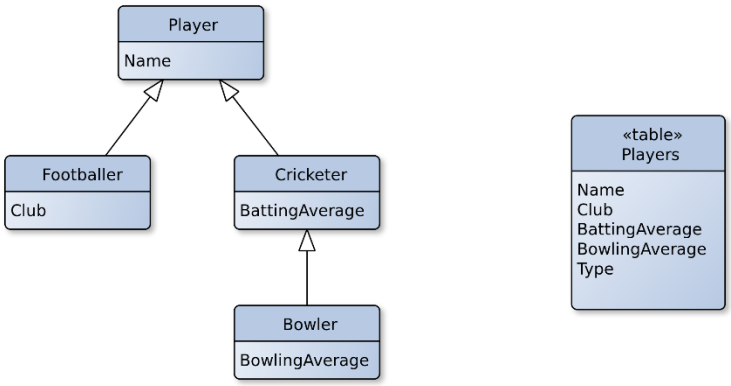
Nachteile

- Ich weiss nicht

Anwendung

Einsatzgebiete sind unter anderem Systeme zur Erfassung geographischer Daten (GIS), bei denen Koordinaten miteinander verknüpft sind oder andere Daten referenzieren. Beispielsweise referenzieren mehrere Koordinaten-Objekte eine Straße: die Koordinaten stehen in Relation mit einem Straßennamen und sind selbst Objekte, die zueinander eine Beziehung haben.

Beispiel



NoSQL

Den Namen „NoSQL“ ist die Abkürzung von „**N**ot **o**nly **SQL**“ was auf Deutsch „nicht nur SQL“ bedeutet. NoSQL-Datenbanken wurden kreiert, um weit verteilte, unstrukturierte Datenbestände bearbeiten und analysieren zu können. In den kommenden Jahren werden daher nicht-relationale Datenbanken aufgrund steigenden Datenflut an Bedeutung gewinnen und mit den relationalen Datenbanken in Konkurrenz treten.

Key-Value-Datenbank hat ein sehr einfaches Datenmodell. Unter einem Schlüssel wird ein Wert gespeichert. Weitere Suchmöglichkeiten gibt es nicht. Die Datenbank ermöglicht eine sehr gute Skalierbarkeit und Performance.

Bei den **Large-Column-Datenbanken** ist das Datenmodell mächtiger und dem relationalen Modell ähnlicher, d.h. es basiert auf einzelnen Tabellen. Auch hier sind viele Spalten und komplexe Tabellen möglich jedoch werden Beziehungen zischen Tabellen nicht unterstützt.

Dokumentenorientierte Datenbanken haben ein flexibles Datenmodell. Sie speichern Dokumente in JSON (JavaScript Object Notation) ab und können so beliebig komplizierte Datenstrukturen abbilden. Dabei gibt es kein Schema, so dass Datensätze in einer Datenbank beliebige Strukturen haben können. Dennoch bieten diese Datenbanken eine gute Skalierbarkeit.

Vorteile

- Skalierbarkeit durch Scale-out (das Hinzufügen von Servern)
- Gute Performance
- Datenstrukturen lassen sich effizient bearbeiten
- Grosse Datenmengen

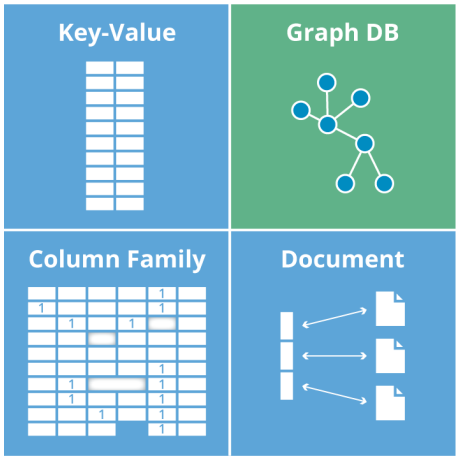
Nachteile

- Beim Scale-out muss der Server in der Lage sein eine Abfrage zu bearbeiten ohne mit anderen Servern zu kommunizieren
- Nicht so Konsistent

Anwendung

Sehr geeignet für grosse Datenvolumen.

Beispiel



Big Data

Der Begriff „Big Data“ bezeichnet die große Menge an strukturierten und unstrukturierten Daten, die Unternehmen Tag für Tag überschwemmen. Nicht die Daten selbst sind so wichtig. Was zählt, ist das, was Unternehmen mit den Daten machen. Große Datenmengen können analysiert werden, um Erkenntnisse zu gewinnen und auf deren Grundlage bessere Entscheidungen zu treffen und das Unternehmen strategisch auszurichten.
Der Begriff „Big Data“ ist zwar relativ neu, das Erfassen und Speichern großer Mengen an Informationen für eine spätere Analyse wird jedoch bereits seit sehr langer Zeit praktiziert. Der Begriff gewann in den frühen 2000er Jahren an Bedeutung, als Branchenanalyst Doug Laney die heute anerkannte Definition von Big Data in seinem 3-V-Modell formulierte:
Volume. Unternehmen sammeln Daten aus einer Vielzahl von Quellen: geschäftlichen Transaktionen, sozialen Medien sowie Sensor- oder Machine-to-Machine-Daten. Früher wäre die Speicherung dieser Daten ein Problem gewesen. Dank neuer Technologien wie Hadoop ist das heute viel einfacher.

Velocity. Die Datenströme bewegen sich in nie da gewesener Geschwindigkeit und müssen zeitnah verarbeitet werden. RFID-Tags, Sensoren und Smart Metering sorgen dafür, dass riesige Datenmengen nahezu in Echtzeit verarbeitet werden müssen.
Variety. Daten fallen in unterschiedlichsten Formaten an – von strukturierten, numerischen Daten aus herkömmlichen Datenbanken bis hin zu unstrukturierten Textdokumenten, E-Mail, Video, Audio, Börsentickerdaten und Finanztransaktionen.
Bei SAS berücksichtigen wir zwei weitere Dimensionen, wenn wir über Big Data nachdenken:
Variabilität. Zusätzlich zu der wachsenden Geschwindigkeit und Vielfalt der Daten kann der Datenfluss sehr unbeständig sein und periodische Spitzen aufweisen. Gibt es einen Trend in den sozialen Medien? Tägliche, saisonale und durch Ereignisse ausgelöste Datenspitzen sind unter Umständen schwer zu bewältigen. Das gilt umso mehr, wenn es sich um unstrukturierte Daten handelt.
Komplexität. Daten stammen heute aus verschiedenen Quellen und das macht es schwierig, sie systemübergreifend zu verknüpfen, anzupassen, zu bereinigen und zu übertragen. Sie müssen sie jedoch verbinden und Beziehungen, Hierarchien und vielfache Datenverknüpfungen korrelieren, da Ihre Daten sonst schnell außer Kontrolle geraten würden.

Beispiels

Quellen

Informationen:

https://de.wikipedia.org/wiki/Datenbankmodell
https://de.wikipedia.org/wiki/Hierarchisches_Datenbankmodell
https://de.wikipedia.org/wiki/Netzwerkdatenbankmodell
https://de.wikipedia.org/wiki/Objektdatenbank
https://www.itwissen.info/Objektrelationale-Datenbank-object-relational-database-.html
https://www.computerweekly.com/de/antwort/Netzwerk-hierarchische-und-nicht-relationale-Datenbankmodelle-im-Vergleich

Bilder:

https://www.datenbanken-verstehen.de/datenbank-grundlagen/datenbankmodell/hierarchisches-datenbankmodell/

https://www.google.com/search?q=relationale+datenbank&rlz=1C1CHBF_deCH856CH856&sxsrf=ACYBGNSUSuEBCCGL4T1duxez8SXKFsFKtg:1570612512281&source=lnms&tbm=isch&sa=X&ved=0ahUKEwjv9P7G647lAhULPVAKHfeMCw0Q_AUIEigB&biw=1600&bih=1057#imgrc=lj5EkXjwsHgbJM: