**Grundlagen der Datenbanksysteme und das relationale Konzept**

Unsere materielle Welt besteht aus Objekten und Informationen. Seit den Anfängen der Menschengeschichte bemüht sich der Mensch diese auf geeigneten Trägern festzuhalten. Jetzt im Zeitalter des digitalen Fortschritts dienen Datenbanken als Speicherort für allerlei Informationen. Kaum eine Webseite kommt heutzutage ohne eines in ihrem Hintergrund agierenden Datenbanksystems aus. Egal ob es um soziale Netzwerke, Online-Shopping, Forschung oder Unternehmenslogistik geht – überall werden riesige Mengen an Informationen gespeichert, organisiert, verarbeitet und nach Außen zur Verfügung gestellt. Dieses Lernheft bringt Ihnen die Grundlagen der faszinierenden Welt von Datenbanksystemen und legt dabei den Schwerpunkt auf das relationale Konzept.

**Probleme bei herkömmlichen Datenspeicherung**

Wozu braucht man eine Datenbank, wenn man doch Daten auch „klassisch“ in Ordnern und Dokumenten ablegen kann bzw. was macht eine Datenbank besser gegenüber gängigen Speichermethoden auf einem Computer oder Smartphone. Ein Einwand könnte lauten: „Ich, als Privatanwender komme ich auch ohne Datenbank aus.“ Ja, es kommt immer auf die Menge. Je größer die Menge an Informationen, desto mehr empfiehlt sich der Einsatz eines „intelligenten“ Datenbanksystems. Macht man es nicht wird es mit der ansteigenden Informationenlast immer unüberschaubarer.

Bei den herkömmlichen Methoden der Datenspeicherung wurden unter anderen folgenden Problemen erkannt:

* **Redundanz**

istmehrfaches Vorhandensein von gleichen Daten an mehreren Stellen.  
**Beispiel**: Zahlreiche Kunden der Firma „Stift & Spaß“ wohnen in der Musterstraße. Die Firma verwaltet ihre die Daten ihrer Kundschaft in einer großen Excel-Tabelle. Bei manchen der Einträge haben sich auch Tippfehler eingeschlichen. So heißt darin die Musterstraße auch Musterstrasse, Musterstr. oder auch ganz falsch Mustersstraße. Eines Tages anfangs Dezember entschließt sich die Stadtverwaltung den sehr eintönig klingenden Namen „Musterstraße“ in Blumenstraße umzubenennen. Es ist Zeit Weihnachtswünsche zu versenden. Durch die Umbenennung sind viele der Adressen falsch. Ein großer manueller Aufwand kommt auf die Angestellten zu, jede Adresse muss gesondert angeschaut und ggfs. auf eine aktuelle mit „Blumenstraße“ geändert werden.  
**Nachteil**: Daten müssen unter hohem Aufwand an mehreren Stellen gewartet werden.

* **Inkonsistenz**

ist ein Zustand, in dem die Daten widersprüchlich oder unvollständig sind, was zu Fehlern oder unerwünschtem Verhalten führt.

**Beispiel**: „Stift & Spaß“ hat noch am Lager 20 Päckchen Druckerpapier. Ein Kunde nimmt zwei Päckchen ab. Durch einen Absturz einer veralteten Software wird der Lagerstand von 18 Päckchen nicht aktualisiert. Gleich gelingt es einem Großkunden im „Stift & Spaß“-Onlineshop 20 Päckchen Druckerpapier zu bestellen, obwohl nur 18 lagernd sind aber das System noch den alten Stand von 20 Päckchen hat.  
**Nachteil**: Fehlerhafte und unzuverlässige Daten, die kaum analysierbar sind.

* **Datenschutzprobleme**

**Beispiel**: Praktisch jeder Angestellte von „Stift & Spaß“ hat einen Zugriff auf die Kundenliste, die in einer Excel-Tabelle gehalten wird und kann es auch auf seinen USB-Stick kopieren.  
**Nachteil:** Risiko der Preisgabe von vertrauenswürdigen und gesetzlich geschützten Daten an Unbefugte.

* **Fehlende Datenunabhängigkeit**

**Beispiel**: Firma „Stift & Spaß“ verwaltet ihre Produkte in einem extra dafür individuell entwickelten Programm. Jedes Produkt verfügt dort über seine Grundeigenschaften, wie Name, Preis, Beschreibung etc. Aufgrund eines neuen Gesetzes sollte nun bei bestimmten Produkttypen ein neues Merkmal mitgespeichert werden. „Stift & Spaß“ beauftragt den Hersteller des Programms mit den notwendigen Anpassungen.   
**Nachteil:** Daten sind gekoppelt an das sie verwaltendes Programm. Wegen fehlender Standardisierung ungewisse Wartungs- und Erweiterungsfrage.

**Datenbanksystem**

Oben beschriebene Probleme wurden nach und nach erkannt und Schritt für Schritt mit den immer besser werdenden Datenbanksystemen gelöst.

Zu einem Datenbanksystem (**DBS**) gehört außer mindestens einer Datenbank (**DB**) auch Software, die man allgemein als Datenbankmanagementsystem (**DBMS**) bezeichnet und durch die Operationen (wie: Einfügen, Abfragen, Aktualisieren, Löschen) an den in der DB liegenden Daten ausgeführt werden können.

Eine Datenbank weist folgende Eigenschaften auf:

* Daten werden entsprechend ihrer natürlichen Eigenschaften und Zusammenhänge aus der realen Welt gespeichert.  
  Beispiel: Ein Kunde kann eine oder mehre Bestellungen aufgeben. Eine Firma interagiert mit Personen – Personen können Mitarbeiter, Lieferanten oder Kunden sein.
* Daten werden in fixen logischen Container-Strukturen aufbewahrt, sind redundanzfrei (wiederholen sich nicht) und geschützt gegen Inkonsistenzen.
* Daten sind gesichert vor unbefugten Zugriffen.

**Kurz**: Eine Datenbank ist eine **strukturierte Sammlung von Daten**, die elektronisch gespeichert werden.

Ein Datenbankmanagementsystem (DBMS) ermöglicht

* Anlage, Änderung und Löschung von Datenbanken und Datenbankobjekten (z.B. Tabellen mit ihren Attributen)
* Einfügen, Ändern und Löschen von Daten
* Jegliche Abfragen und Analysen der Daten
* Verwaltung von Benutzern, Rollen, Zugriffen und Zugriffsrechten
* Sicherung (Backup) von Daten und deren Wiederherstellung
* noch vieles mehr

**Kurz**: **Datenbankmanagementsystem** ist eine Software zur Verwaltung und Nutzung der Datenbank (z. B. MS SQL Server Management System).

**Datenbanktypen**

**Hierarchische Datenbanken**

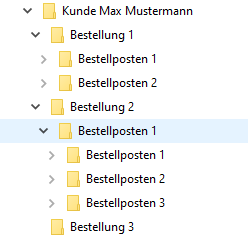
Das hierarchische Datenmodell dient zur Verarbeitung unterschiedlich langer Datensätze. Die Daten werden in kleinere, logisch zusammengehörige Gruppen unterteilt. Diese Gruppen bilden die Knoten einer baumartigen Hierarchie – ähnlich der Struktur eines Stammbaumes.

Jeder untergeordnete Knoten hängt von einem übergeordneten Knoten ab. Die Beziehungen folgen dem Prinzip einer Vater-Sohn-Struktur. Ein Vaterknoten kann mehrere Kindknoten haben, aber jeder Kindknoten hat nur einen Vater. Die Hierarchie benötigt immer einen Wurzelknoten.

**Beispiel**

Ein Kunde kann mehrere Bestellungen tätigen. Jede Bestellung kann mehrere Bestellposten enthalten.

Ein Beispiel ist eine Kundenverwaltung. Ein Kunde kann mehrere Bestellungen aufgeben. Jede Bestellung kann mehrere Positionen enthalten. Die Daten zu Kunden, Bestellungen und Positionen sind die Knoten im Hierarchiebaum. Für jeden Knoten wird ein Datensatz angelegt. Auch ein Dateienexplorer funktioniert nach der Art einer hierarchischen Datenbank.



Vorteile:

* **Schneller Datenzugriff**  
  Elementfindung ist dank der klaren sehr effizient.
* **Einfache Datenintegrität**  
  Die festen Beziehungen zwischen den Knoten gewährleisten konsistente Daten.
* **Klares Schema**  
  Das baumartige Modell ist leicht verständlich.

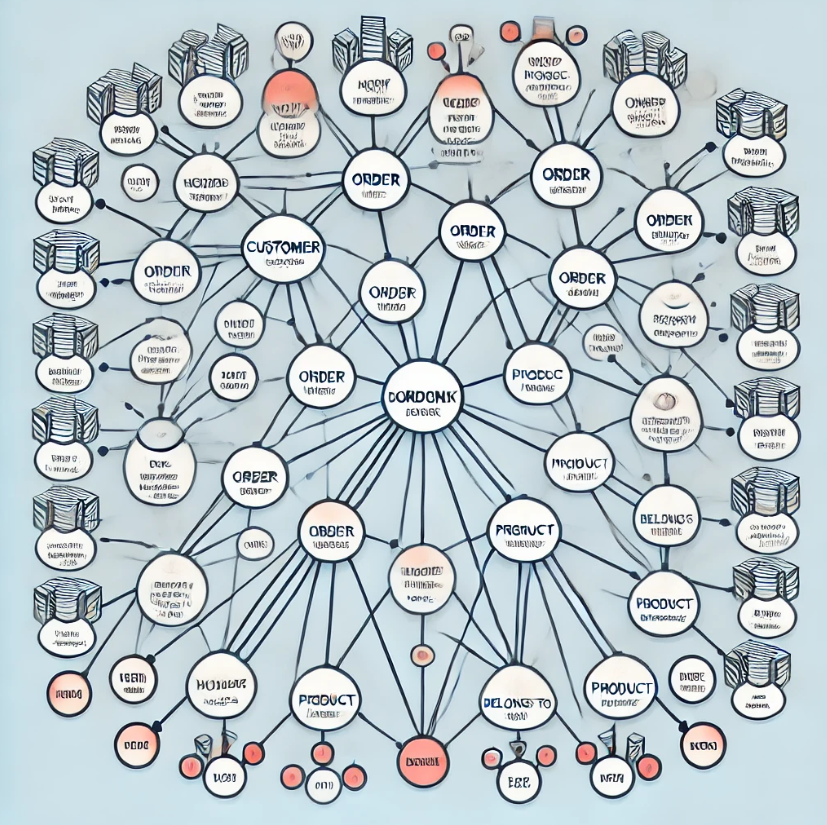
Nachteile:

* **Unflexible Struktur:**  
  Änderungen in der Hierarchie sind aufwendig. Das Modell ist starr und wenig anpassungsfähig.
* **Komplexe Verwaltung:**  
  Der Zugriff auf Daten außerhalb der Hierarchie ist schwierig und erfordert oft zusätzliche Programmierung.
* **Redundante Daten:**  
  Daten, die in mehreren Hierarchien benötigt werden, müssen oft dupliziert werden, da Querverweise fehlen.
* **Schlechte Skalierbarkeit:**  
  Für große und komplexe Datenstrukturen ist das Modell ineffizient und schwer wartbar.
* **Abhängigkeit vom Wurzelknoten:**  
  Fällt der Wurzelknoten aus, ist die gesamte Struktur unbrauchbar.

Ein bekanntes Datenbanksystem, das mit dem hierarchischen Datenmodell arbeitet, ist **IBM Information Management System (IMS)**. IMS wurde in den Jahren 1966 bis 1968 für die Stücklistenverwaltung des Apollo-Mondprogramms entwickelt.

**Netzwerkdatenbanken**

Funktionsweise: Daten werden als Netz von Knoten und Verbindungen organisiert. Das Netzwerkdatenmodell erweitert das hierarchische Modell um dort fehlende Beziehungen.



**Vorteile**

* **Flexible Beziehungen:**  
  Es ermöglicht komplexe Beziehungen (Verweise) zwischen den Daten.
* **Schnelle Datenzugriffe:**  
  Direkte Verweise zwischen Datensätzen sorgen für effiziente Findung von Elementen.
* **Weniger Redundanz:**  
  Dank den Verweisen brauchen die Daten seltener vervielfacht werden.

**Nachteile**

* **Komplexe Verwaltung:**  
  Verweise müssen manuell gepflegt werden, was die Verwaltung und Wartung der Datenbank erheblich erschwert.
* **Erfordert genaue Pfadkenntnisse:**  
  Der Benutzer muss den genauen Navigationspfad kennen, um Daten abzurufen.
* **Eingeschränkte Flexibilität:**  
  Bei neuen Anforderungen und Änderungen der Datenstruktur ist das Modell weniger flexibel als relationale Datenbanken.
* **Steile Lernkurve:**  
  Die allgemeine Komplexität des Systems erschwert das Verständnis und somit die Nutzung.

Ein bekanntes Beispiel für ein Netzwerkdatenbanksystem ist das CODASYL DBTG. Es wurde in den 1960er Jahren entwickelt und sein Einsatzgebiet war vorrangig in Finanzinstitutionen und Versicherungsgesellschaften, wo die Daten in komplexen und stark miteinander verknüpften Netzwerken gespeichert werden mussten. Trotz seines Erfolges verlor das CODASYL DBTG-System in den 1980er Jahren zugunsten relationaler Datenbanksysteme zunehmend an Popularität.

**Objektorientierte Datenbanken**

Daten werden als Objekte (ähnlich wie in der objektorientierten Programmierung) gespeichert. Jedes Objekt kann Attribute und Funktionen haben, die auf diese Daten zugreifen.

Beispiel: Objekt Produkt enthält Attribute Name, Hersteller, Preis und Lagerstand und Methode GibLagerstand() (Gibt die Verfügbarkeit des Produkts als Stückzahl zurück).

Objektorientierte Datenbanken starke **Vorteile** in Bezug auf die Abbildung komplexer Datenstrukturen und die nahtlose Integration mit objektorientierten Programmiersprachen, haben jedoch auch Nachteile in Bezug auf Komplexität und Leistung bei sehr großen Datenmengen. Daher eignen sie sich eher nur für spezifische Anwendungen, bei denen diese Vorteile besonders zur Geltung kommen.

Als bekannter Vertreter auf dem Markt der objektorientieren Datenbanksysteme gilt ObjectDB von der Firma ObjectDB Software.

**NoSQL-Datenmodelle**

NoSQL-Datenbanken sind eine beliebte Alternative zur Objektdatenbank. Sie verfolgen einen nicht-relationalen Ansatz, das heißt sie versuchen Verbindungen zwischen den Daten weitgehend zu vermeiden.

**Vorteile**

* **Flexibilität**

Keine festen Strukturen: Struktur kann sich dynamisch ändern.

Ideal für unstrukturierte oder teilweise strukturierte Daten (z. B. JSON, XML).

* **Erweiterbarkeit**

Unterstützen große Datenmengen und lassen sich einfach über mehrere Server hinweg skalieren.

* **Hohe Leistung und Verfügbarkeit**

Geeignet für Echtzeitanwendungen wie Streaming-Daten oder Web-Anwendungen, da schnelle Reaktion und Verfügbarkeit meistens möglich.

**Nachteile**

* **Konsistenzprobleme**

Der Preis für die hohe Leistung drückt sich als „eventual consistency" (Daten sind nicht sofort konsistent) aus.

* **Fehlen von Standards**

Es gibt keine einheitliche Abfragesprache wie SQL.

* **Eingeschränkte Funktionalitäten**

Keine oder begrenzte Unterstützung für komplexe Transaktionen. Datenabfragen und Analysen können komplizierter sein als bei relationalen Datenbanken.

* **Lernkurve**

Junge Technologie, steckt noch in den Kinderschuhen. Entwickler und Administratoren müssen neue Konzepte und Abfragesprachen lernen. Weniger verbreitete Expertise im Vergleich zu relationalen Datenbanken.

* **Speicherverbrauch**

Einige NoSQL-Datenbanken haben eine höhere Speicheranforderung durch redundante Daten (sich wiederholende Daten).

**Azure Cosmos DB** ist ein vollständig verwalteter, global verteilter NoSQL-Datenbankdienst von Microsoft Azure, der für moderne Anwendungen entwickelt wurde. Mit der DBMS -artigen SQL-Schnittstelle können Konsumierer Objekte erstellen, aktualisieren und löschen. Daten können mit einem SQL-Dialekt abgefragt werden.

**Relationale Datenbanken**

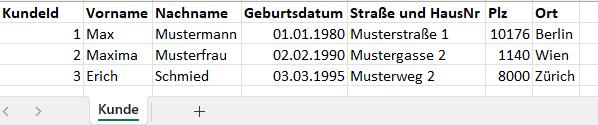
Das in den 1970er Jahren von Edgar F. Codd entwickelte relationale Datenbankmodell wird heute am meisten verwendet. Die Daten werden in Tabellenform in sogenannten **Relationen** gespeichert. Zwischen den Relationen können logische Verbindungen (Beziehungen) definiert werden. Das relationale Datenbankmodell genießt eine breite Unterstützung in vielen bewährten Datenbanksystemen. **Microsoft SQL Server**, MySQL, PostgreSQL, Oracle Database oder SQLite setzen seit Jahrzehnten darauf. Und das nicht ohne Grund, denn eine gut designte relationale Datenbank verfügt sind stets konsistente Daten, die kaum Redundanzen vorweisen und die mit der standardisierten Abfragesprache SQL abgefragt und geändert werden können.

**Das relationale Konzept und dessen Umsetzung**

**Konzept**

**Vergleich Blatt in Tabellenkalkulation – Tabelle in Datenbank**

Wahrscheinlich kennen Sie schon das Konzept einer Tabellenkalkulation oder haben Sie sogar damit bearbeitet und können hier sofort von dieser Vorerfahrung profitieren. Es ist sehr einfach: eine Tabelle besteht aus Zeilen, Spalten und Zellen.



Auch in einer relationalen Datenbank werden Daten in Tabellen (Relationen) mit Zeilen (Datensätze), Spalten (Attributen) und Zellen (Attributwerten) gespeichert. Sowohl in einer Tabellenkalkulation als auch in einer Datenbanktabelle werden oft Zeilen (Datensätze) mit einem oder einem zusammengesetzten, speziellem Attributwert versehen, mit dem Mann die jeweilige Zeile bzw. den jeweiligen Datensatz eindeutig identifizieren kann.

In dem hier dargestellten Ausschnitt aus einer Tabellenkalkulation ist es die Spalte „PersonenId“.

Im Kontext einer relationalen Datenbank nennt den derart gezeichneten Attribut „Primärschlüssel“.

