

Datenbanksystem und Datenbank-Entwurf

Skript

Inhaltsverzeichnis

1	Grundlagen zum Datenbanksystem (DBS)	2
1.1	Komponenten eines Datenbanksystems (DBS)	2
1.2	Die 9 Codd'schen Regeln für ein leistungsstarkes DBS	2
1.3	ANSI-SPARC-Architektur	3
2	Relationale Datenbank	4
2.1	Primärschlüssel	4
2.2	Datenbank-Tabelle mit Fachbegriffen	4
3	Prozess der Datenbank-Erstellung	5
4	Entity-Relationship Modell (ERM) nach Chen	5
4.1	Aufbau und Interpretation des Entity Relationship Modells	6
5	Übersetzung des ER Modells in Datenbank-Tabellen	6
6	Normalisierung und 1-3 Normalformen (NF)	7
6.1	Überblick: Normalformen 1-3	7
6.2	Die 1. Normalform	7
6.3	Die 2. Normalform	8
6.4	Die 3. Normalform	8
7	Referenzielle Integrität	8

1 Grundlagen zum Datenbanksystem (DBS)

1.1 Komponenten eines Datenbanksystems (DBS)

Ein Datenbanksystem (DBS) ist ein System zur Beschreibung, Speicherung und Wiedergewinnung umfangreicher Datenmengen, das von mehreren Anwendungsprogrammen oder Anwendern benutzt wird.

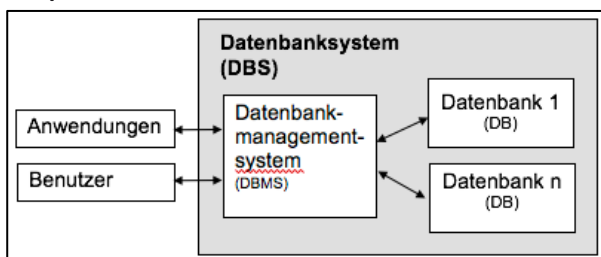
Ein Datenbanksystem (DBS) besteht aus folgenden Komponenten:

- **dem Datenbankmanagementsystem (DBMS):**
Dies repräsentiert das zentrale Softwarepaket des DBS. Aufgaben eines DBMS sind:
 - Administration der Datenbanken,
 - Speicherung, Ausgabe und Bearbeitung von Daten
 - Zugriffskontrolle (d.h. die Autorisationsprüfung und die Behandlung gleichzeitiger Zugriffe verschiedener Benutzer (Synchronisation),
 - Bereitstellen einer Datensicherungsmöglichkeit für den Fall von Systemausfällen.
- **den Datenbanken:**
Diese beheimaten die eigentlichen Daten. Es werden jeweils die Daten eines Anwendungsbereichs in einer Datenbank gespeichert. Das DBMS kann grundsätzlich mehrere Datenbanken steuern.

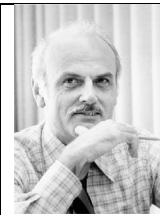
Das DBMS benötigt selber eigene sog. Systemdaten. Dieser Datenbestand wird als Data Dictionary bezeichnet und stellt eine Art Inhaltsverzeichnis des DBS dar. Es enthält u. a. die Auflistung und Beschreibung der einzelnen Datenbanken, z. B. welche Tabellen, Spalten usw. vorhanden sind. Das Data Dictionary ist in der Regel eine eigene Datenbank innerhalb des DBS.

Zur Arbeit mit einem DBS werden Datenbanksprachen verwendet, die vom DBMS verarbeitet werden. Als Standard-Datenbanksprache wird **SQL** (Structured Query Language) verwendet.

Komponenten des DBS in der Übersicht:



1.2 Die 9 Codd'schen Regeln für ein leistungsstarkes DBS

	<p>Edgar F. Codd (1923-2003)</p> <p>A Relational Model of Data for Large Shared Data Banks in: Communications of the ACM, Volume 13, Issue 6 (June 1970) Pages: 377 - 387 (1970)</p> <p>Turing Award (1981) = Nobelpreis der Informatik https://de.wikipedia.org/wiki/Edgar_F._Codd</p>
------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

1 Daten-Integration: einheitliche, nichtredundante und damit konsistente Datenverwaltung

Unterschiedliche Anwendungen speichern ihre Daten zentral zusammen. Daten werden nicht **redundant** gespeichert (d.h. gleiche Daten werden mehrfach bzw. doppelt gespeichert). Durch redundante Daten, besteht die Gefahr, dass inkonsistente Daten erzeugt werden, weil Änderung an einer Version der Daten nicht komplett, korrekt und zeitgleich auf alle anderen Versionen durchzuführen sind (Konsistente Daten = Widerspruchsfreie Daten).

2 Daten-Operationen: Speichern, Suchen, Ändern, Löschen, Anlegen von Daten

Das DBMS ermöglicht die Bearbeitung des Datenbestandes und setzt diese optimiert um. Die benötigten Kommandos können sowohl vom Benutzer direkt eingegeben werden ("ad hoc") als auch im Programm (Quellcode) integriert ("embedded") werden. Zur Kommunikation zwischen Benutzer/Programm und DBMS wird die Datenbanksprache SQL (SQL = Structured Query Language) verwendet.

3 Daten-Katalog: Zugriffe auf Datenbankbeschreibung im "Data Dictionary"

Die Organisation der Datenhaltung muss übersichtlich und leicht zugänglich sein.

4 Benutzersichten: unterschiedliche Sichten auf Datenbankausschnitte

Daten können beliebig zusammengestellt werden. Eine Kundenmaske kann z.B. je nach gewünschten Kontext die Vertragsdaten und die Adressdaten des Kunden anzeigen oder z. B. nur seine Adressdaten.

5 Konsistenzüberwachung: Gewährleistung der Korrektheit des Datenbankinhaltes

Datenbestände müssen widerspruchsfrei gehalten werden. Z. B. darf in einer Produktdatenbank keine Produktnummer zweimal vergeben werden. Das DBMS hat die Aufgabe, derartige Fehler zu verhindern und die Konsistenz des Datenbestandes zu überwachen.

6 Datenschutz: Ausschluss unautorisierter Zugriffe

Ein DBS sollte ein Berechtigungssystem besitzen.

7 Transaktionen: mehrere Operationen als Funktionseinheit

Es können Blöcke von Aktionen auf Daten definiert werden. Nur wenn alle einzelnen Aktionen des Blockes fehlerfrei ausgeführt werden können, werden diese auch tatsächlich durchgeführt (commit). Beim Auftreten eines Fehlers werden schon durchgeführte Aktionen "zurückgefahren" (rollback). Beispiel für eine Blockaktion: Überweisung eines Geldbetrages: beide Aktionen, das Abbuchen eines Betrages von einem Konto A und das Zufügen dieses Betrages auf das Konto B müssen durchgeführt werden, sonst ergeben sich falsche Kontostände.

8 Synchronisation: parallele Transaktionen koordinieren

Ein DBS muss mehrbenutzerfähig sein und konkurrierende Zugriffe ermöglichen und steuern.

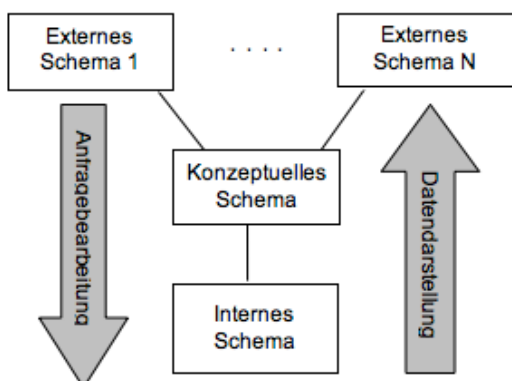
9 Datensicherung : Wiederherstellung von Daten nach Systemfehlern

Spezielle Funktionen erlauben das Wiederherstellen von konsistenten Datenbeständen zu jedem gewünschten Zeitpunkt.

1.3 ANSI-SPARC-Architektur

Ein zentrales Ziel eines DBS ist, eine (weitgehende) Unabhängigkeit von Daten mit ihren arbeitenden Programmen/Benutzern zu erreichen. Benutzer/Programme brauchen keine Kenntnisse über Ort und Struktur der gespeicherten Daten.

Modifikationen von Speicherstrukturen haben keine Auswirkungen auf vorhandene Anwendungsprogramme. Dies bezeichnet man **Datenunabhängigkeit**. Realisiert wird dies durch eine sogenannte ANSI-SPARC Architektur (von 1975):



Die **ANSI-SPARC-Architektur**, auch 3-Schichten-Architektur, beschreibt das grundlegenden Konzept eines relationalen Datenbanksystems.

Die drei Ebenen sind:

- Die externe Ebene, die den Benutzern und Anwendungen individuelle Benutzersichten bereitstellt, z. B. Bildschirmmasken.
- Die konzeptuelle Ebene, in der beschrieben wird, **welche** Daten in der Datenbank gespeichert sind sowie deren Beziehungen untereinander (s. ERM oder RDBM).
- Die interne Ebene, die die physische Sicht der Datenbank im Computer darstellt. In ihr wird beschrieben, **wie** und **wo** die Daten in der Datenbank gespeichert werden.

Die Vorteile des 3-Ebenen-Modells liegen in der **physischen und logischen Datenunabhängigkeit**.

Physische Datenunabhängigkeit:

Anwenderprogramme benötigen keine Kenntnisse über die interne Speicherung der Daten (Trennung von externer und interner Ebene). Programme/Benutzer benutzen nur die Datenbegriffe aus der Konzeptionellen Schicht (z.B. eine Kundentabelle). Wo diese Kundentabelle im Filesystem liegt ist irrelevant. So hat z.B. eine Erweiterung von Speicherstrukturen keine Auswirkungen auf Anwendungsprogramme.

Logische Datenunabhängigkeit:

Ein Datenbestand kann durch verschiedene Anwendungen unterschiedlichen dargestellt werden (Trennung externer von konzeptioneller Ebene). So kann z. B. die Kundendatenbank problemlos um weitere Tabellen erweitert werden ohne dass bestehende Anwendungen, die die Kundendatenbank benutzen, davon betroffen sind.

2 Relationale Datenbank

Die Grundlage für relationale Datenbanken wurde 1970 von dem Mathematiker E.F. Codd entwickelt. Eine Datenbank besteht demnach aus einer Menge miteinander in Beziehung stehender Tabellen. Der Grundbaustein einer relationalen Datenbank ist die **Tabelle** (Tabelle = Relation) mit einer bestimmten Anzahl von Spalten und Zeilen, in denen logisch zusammengehörige Daten gespeichert werden.

Eigenschaften einer Tabelle in einer relationalen Datenbank:

- Eine Tabelle besitzt einen eindeutigen Tabellennamen.
- Die Bezeichnung der Attribute in der Tabelle sind eindeutig.
- Eines der Attribute (oder eine bestimmte Kombination mehrerer Attribute) identifiziert eindeutig einen Datensatz. Dieses Merkmal bzw. diese Kombination bezeichnet man als Primärschlüssel (s. unten).
- Jedem Attribut wird ein bestimmter Datentyp zugeordnet. Dieser legt Speichergröße und den Wertebereich (Wertebereich = Domain) des Attributs fest. Hierfür stellt jedes DBS u.a. verschiedene Zahlentypen, Datumstypen und Zeichenkettentypen zur Verfügung.
- Die Reihenfolge der Attribute hat keine Bedeutung.

Die Angabe dieser Eigenschaften in der Form: Tabellename (Spalte1: Datentyp1, Spalte2: Datentyp2,...) wird auch als **Schema der Tabelle** bezeichnet (Hinweis: Primärschlüssel ist unterstrichen).

Datenbanksysteme, die aus relationalen Datenbanken bestehen, sind heute am weitesten verbreitet. Beispiele für relationale Datenbanksysteme sind z. B. MS Access, MySQL oder Oracle.

2.1 Primärschlüssel

Um Daten in einer Datenmenge eindeutig identifizieren zu können, darf es keine zwei Datensätze geben, die exakt gleich sind; sie müssen sich mindestens in einem Attributwert unterscheiden. Um dies zu gewährleisten wird ein sog. Primärschlüssel gefordert.

Definition: Primärschlüssel

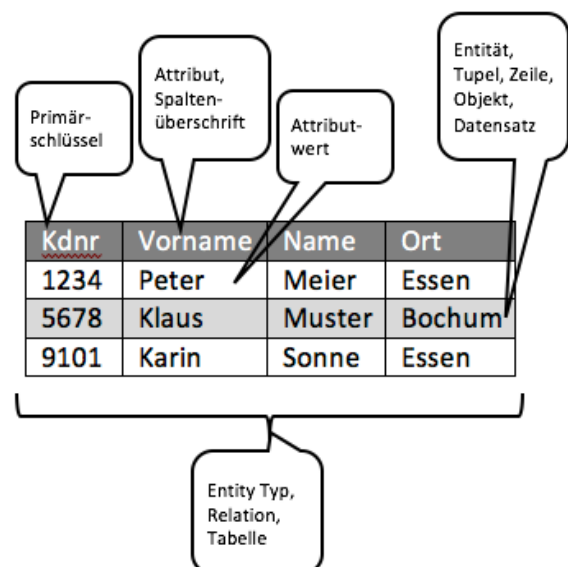
Ein Attribut oder eine minimale Kombination aus Attributen, die jeden Datensatz eindeutig identifiziert.

Minimal bedeutet in diesem Kontext, dass sobald ein Attribut weggenommen wird, die Eindeutigkeit verloren geht. Ist in der Menge der Attribute kein

geeigneter Primärschlüssel-Kandidat enthalten, kann ein zusätzliches (künstliches) Attribut ergänzt werden (z. B. KundenNr wenn Kundendaten gespeichert werden sollen).

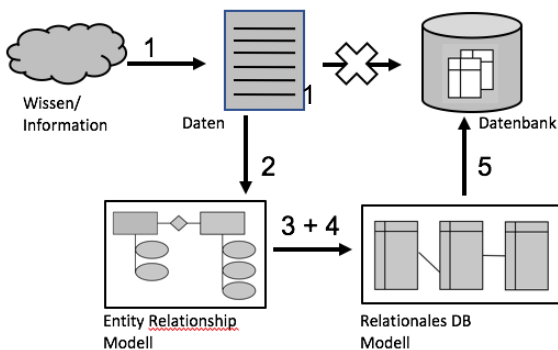
Beim Erstellen einer Datenbank-Tabelle wird automatisch zusätzlich ein Index über alle Primärschlüssel erstellt. Mithilfe des Index ist ein schneller direkter Zugriff auf den gewünschten Datensatz möglich. Dies kann man sich vorstellen wie bei einem Buch. Wenn Sie eine bestimmte Stelle in einem Buch suchen, ist es schneller hinten im Index die Seitenzahl zu ermitteln, als von Beginn an jede Seite nach dem Gewünschten zu durchsuchen.

2.2 Datenbank-Tabelle mit Fachbegriffen



* Relation = Menge aller Zeilen, Entity Typ = Tabellenstruktur. Die 3 Begriffe Tabelle, Relation und Entity Typ können allerdings in erster Näherung gleichbedeutend verwendet werden.

3 Prozess der Datenbank-Erstellung



Die Entwicklung einer Datenbank vollzieht sich in 5 Teilschritten:

- Datenanalyse**
Die zu speichernden Daten müssen aus dem Pflichtenheft (Idealfall), aus alten Anwendungen oder Kundeninterviews ermittelt werden. Neben der Auflistung der Daten sind auch weitere Informationen (z. B. Datenvolumen, Zugriffsbeschränkungen usw.) wichtig. Diese werden jedoch erst in der Implementierungsphase wichtig.
- Datenbankentwurf in Form eines Entity Relationship Modells (ERM)**
 - Datengruppen identifizieren und mit einem Oberbegriff versehen (Entitätstypen festlegen).
 - Primärschlüssel für jeden Entitätstyp festlegen.
 - Beziehungen zwischen den Entitätstypen festlegen.
- Transformation des ERM in ein Relationales Datenbank Modell (RDBM)**
- Normalisierung des Modells testen anhand der Normalformen 1-3.**
- Implementierung der Datenbank mit dem gewünschten Datenbanksystem.**

4 Entity-Relationship Modell (ERM) nach Chen

Der Schwerpunkt bei diesem Modell liegt auf den Daten und ihren Beziehungen. Informationen, die für eine spätere Implementierung wichtig sind, wie z.B. welche Datentypen den Daten zuzuordnen sind, spielen keine Rolle. Dieses konzeptuelles Modell ist im Gegensatz zu dem "späteren" Relationalen Modell (logisches Modell) DBS unabhängig.

Ein Beispiel für ein konzeptuelles Modell ist das so genannte **Entity-Relationship-Modell** (deutsch „Gegenstands-Beziehungs-Modell“). Die Notation erfolgt nach Peter Chen.

Element (Definition)	Symbol (mit Beispiel)
Entitätstyp Oberbegriff (Kategorie) von zusammengehörigen Attributen. Struktur von Datensätzen, die sich durch gleiche Attribute beschreiben lassen.	Rechteck: <div style="border: 1px solid black; padding: 5px; text-align: center;">Mitarbeiter</div> Name in Singular-Form
Attribut Eigenschaft	Ellipse: <div style="border: 1px solid black; border-radius: 50%; padding: 5px; text-align: center;">Name</div>
Primärschlüssel Ein Attribut oder eine minimale Kombination von Attributen, die jeden Datensatz eindeutig identifiziert.	Ellipse mit unterstrichenem Namen <div style="border: 1px solid black; border-radius: 50%; padding: 5px; text-align: center;"><u>PersNr</u></div>
Beziehung (mit Kardinalität = Anzahl der an der Beziehung beteiligten Objekte) 1:1 (eins zu eins Beziehung) Jeder Datensatz eines Entitätstyps Mitarbeiter wird genau einem Datensatz des Entitätstyps Ausweis zugeordnet und umgekehrt. 1:N (eins zu n Beziehung) Ein Datensatz von Mitarbeiter kann mit mehreren Datensätzen von Beurteilung verbunden sein. Umgekehrt ist aber ein Datensatz einer Beurteilung genau mit einem Datensatz von Mitarbeiter verbunden. N:M (n zu m Beziehung) Jedem Datensatz eines Entitätstyps Mitarbeiter können mehrere Datensätze des Entitätstyps Projekte zugeordnet werden und umgekehrt.	<div style="display: flex; align-items: center; justify-content: space-around;"> <div style="border: 1px solid black; padding: 5px;">Mitarbeiter</div> <div style="text-align: center;"> 1 hat 1 </div> <div style="border: 1px solid black; padding: 5px;">Ausweis</div> </div> <div style="display: flex; align-items: center; justify-content: space-around;"> <div style="border: 1px solid black; padding: 5px;">Mitarbeiter</div> <div style="text-align: center;"> 1 hat N </div> <div style="border: 1px solid black; padding: 5px;">Beurteilung</div> </div> <div style="display: flex; align-items: center; justify-content: space-around;"> <div style="border: 1px solid black; padding: 5px;">Mitarbeiter</div> <div style="text-align: center;"> N hat M </div> <div style="border: 1px solid black; padding: 5px;">Projekt</div> </div> Name der Kardinalität: z. B. hat oder gehört zu

4.1 Aufbau und Interpretation des Entity Relationship Modells

Regeln zur Modellierung:

- Entitätstypen werden immer über Beziehungen miteinander verknüpft. Mehrere Beziehungen ausgehend von einem Entitätstyp sind möglich.
- Jeder Entitätstyp muss ein Primärschlüssel-Attribut enthalten. Weitere Attribute sind möglich. Attribute werden über Linien mit dem Entitätstyp verbunden, zu dem sie gehören.
- (Nur) zu einer N:M Beziehung können zusätzlich Attribute gehören, die die Beziehung charakterisieren. Diese Attribute sind direkt mit der Raute zu verbinden.

Beispiel für ein Entity-Relationship Modell:

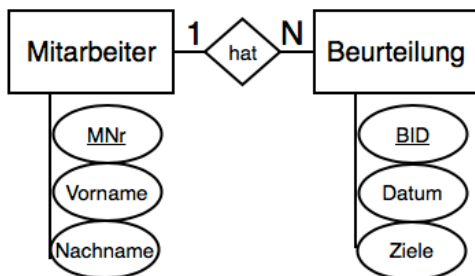


Tabelle von Mitarbeiter (mit Beispiel-Datensätze):

Mitarbeiter		
MNr	Vorname	Nachname
12	Karin	Müller
1456	Peter	Schmidt

5 Übersetzung des ER Modells in Datenbank-Tabellen

Beim Anlegen der Datenbank aus dem ERM muss entschieden werden wie die Beziehungen umgesetzt werden sollen. Diese werden mit Hilfe von Fremdschlüsseln abgebildet.

Definition: Fremdschlüssel (FK)

Ein Fremdschlüssel ist ein Primärschlüssel (PK) aus einer anderen Tabelle und dient zur Speicherung von Beziehungen zwischen Datensätzen.

Wie diese zusätzlichen Beziehungsattribute (Fremdschlüssel) unterzubringen sind und wie damit das relationale Datenbankmodell entsteht, zeigt die folgende Tabelle.

Hinweis: Zur Vereinfachung der folgenden Grafiken sind bei den Modellen außer den Primärschlüsseln keine weiteren Attribute angegeben. Grundsätzlich werden alle anderen Attribute eines Entity Typs in der gleichnamigen Tabelle übernommen.

1 : N Übersetzung	
<p>Pro Entitätstyp wird eine Tabelle erstellt.</p> <p>Die N-Tabelle erhält den Primärschlüssel der 1-Tabelle als zusätzliches Attribut. Dieses Attribut heißt Fremdschlüssel.</p> <p>Die Verbindungslinie wird zwischen den PK und dem dazugehörigen FK gezogen. Bei der Beschriftung bekommt die PK Seite immer die 1 und die FK Seite das N.</p>	
N : M Übersetzung	
<p>Pro Entitätstyp wird eine Tabelle erstellt.</p> <p>Eine zusätzliche Beziehungstabelle wird erstellt. Name frei wählbar. Diese bekommt die beiden Primärschlüssel der N- und M-Tabelle als Fremdschlüssel.</p> <p>Der Primärschlüssel der Beziehungstabelle setzt sich entweder aus der Kombination der beiden Fremdschlüssel zusammen oder es wird ein künstlicher Primärschlüssel zugefügt (z. B. eine ID).</p> <p>Die Beziehungstabelle kann weitere fachliche Attribute enthalten.</p>	
1 : 1 Übersetzung	
<p>Möglichkeit 1: Aus beiden Entitätstypen wird eine gemeinsame Tabelle mit allen Attributen erstellt. Einer der beiden Primärschlüsselkandidaten wird Primärschlüssel.</p> <p>Möglichkeit 2: Pro Entitätstyp wird eine Tabelle erstellt. Der Primärschlüssel einer Tabelle wird als Fremdschlüssel in der anderen Tabelle abgebildet. Welche Tabelle den FK bekommt ist frei wählbar.</p>	
PK = Primärschlüssel, FK = Fremdschlüssel	

6 Normalisierung und 1-3 Normalformen (NF)

Im Regelfall sind die Tabellen einer Datenbank noch nicht optimiert. Es besteht die Gefahr, dass Abhängigkeiten zwischen Attributen zu Redundanzen führen. Dies kann beim Betrieb der Datenbank zu inkonsistenten Daten führen. Damit eine dauerhafte stabile Datenstruktur entsteht, hat sich als methodisches Vorgehen die sog. Normalisierung mit Hilfe von 1-3 Normalformen etabliert.

Definition Normalisierung:

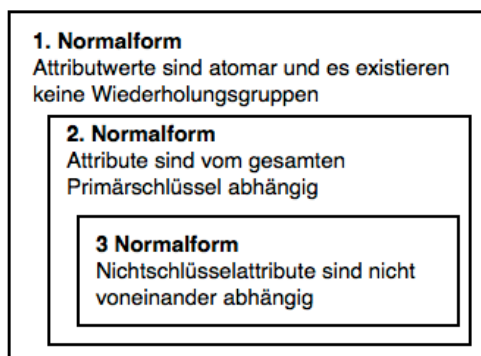
Unter Normalisierung versteht man die Aufteilung von Daten auf mehrere Tabellen unter Berücksichtigung bestimmter Regeln - den Normalformen.

Ziele der Normalisierung

- Erzeugen einer flexiblen, stabilen Datenbankstruktur, bei der Veränderungen leicht durchzuführen sind.
- Vermeiden von Regelwidrigkeiten (Anomalien) beim Ändern (=Änderungsanomalie), Löschen (=Löchanomalie) und Einfügen (=Einfügeanomalie) von Datensätzen.
- Möglichst geringe Redundanz und somit geringer Speicherplatzbedarf und geringe Gefahr von Dateninkonsistenzen.

6.1 Überblick: Normalformen 1-3

Die Tabellen werden mit Hilfe der Normalformen überprüft. Die Normalformen werden immer spezieller und bauen aufeinander auf, d. h. man prüft zunächst ob die 1. Normalform erfüllt ist, danach ob die 2. Normalform erfüllt ist usw.. Die in der Theorie existierende 4 und 5 Normalform wird in der Praxis nicht berücksichtigt. Erst wenn alle Tabellen eines Datenbank-Entwurfs die 3. Normalform erfüllen, wird dieser Entwurf in der Praxis umgesetzt. Die Normalformen stellen daher ein wichtiges Qualitätskriterium für eine Datenbank-Entwurf dar.



6.2 Die 1. Normalform

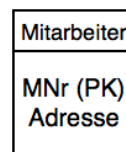
Definition 1. Normalform (1. NF):

Alle Attributwerte müssen atomar sein, d. h. die Attributwerte dürfen sich nicht weiter sinnvoll unterteilen lassen.

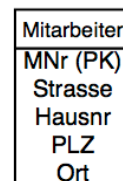
Wiederholungsgruppen sind nicht erlaubt.

Beispiele zur 1. Normalform:

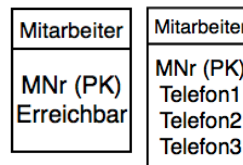
Problem: Adresse:
Es ist kein direkter und damit schneller Zugriff auf die Einzel-Daten z.B. bei einer Suche möglich.



Lösung:
Aufspaltung in einzelne Spalten:

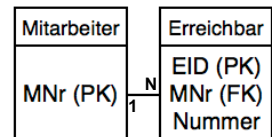


Problem: Auflistung von gleichen Daten z. B. Telefonnr entweder in einer Spalte (s. Erreichbar) oder mehreren Spalten (sog. Wiederholungsgruppen).
Dadurch ist entweder ein schneller Zugriff auf Einzel-Daten erschwert oder es ist unklar wie viele Spalten zu erstellen sind.



Daten Erreichbar: z.B. 12345, 43435, 565656

Lösung:
Zusätzliche Tabelle, die 1:N verknüpft wird.



Hinweise zur 1. Normalform:

In der Praxis finden sich gelegentlich Ausnahmen (dies wird auch als Denormalisierung bezeichnet). Diese bergen allerdings Risiken und sind daher gut zu begründen. Beispiel: Häufig findet man, dass Straße und Hausnr. zusammen verbleiben und nicht als 2 getrennte Attribute abgebildet werden. Dies ist dann sinnvoll, wenn auf die getrennten Informationen seitens der Anwendung (z. B. beim Suchen) definitiv nicht zugegriffen wird.

Ein Attribut, das eine Datumsangabe enthält (z. B. ein Geburtstag) wird nicht in Tag, Monat und Jahr aufgeteilt. Grund hierfür ist, dass bei diesem Attribut ein spezieller Datentyp verwendet wird (z. B. der Datentyp: date), mithilfe dessen jederzeit ein direkter Zugriff (z.B. auf den Tag) ermöglicht wird.

6.3 Die 2. Normalform

Definition 2. Normalform (2. NF)

Die Tabelle befindet sich in der 1. Normalform und alle Attribute hängen vom gesamten Primärschlüssel ab.*

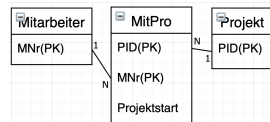
* Abhängigkeit wird auch als volle funktionale Abhängigkeit bezeichnet

Tabellen, die in der 1. Normalform sind, sind automatisch in der 2. Normalform, wenn ihr Primärschlüssel nicht aus mehreren Attributen zusammengesetzt ist (kein Kombinationsschlüssel vorhanden ist). Die zweite Normalform wird also auf Tabellen angewendet, die einen zusammengesetzten Primärschlüssel (Kombinationsschlüssel) haben.

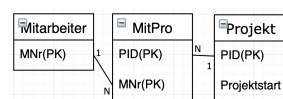
Im Focus der Überprüfung stehen ausschließlich Nicht-schlüsselattribute in Beziehungstabellen.

Beispiel zur 2. Normalform:

Problem: Projektstart ist nur vom Projekt abhängig und nicht von Mitarbeiter.



Lösung: andere Tabelle, Attribut Projektstart gehört in die Projekt Tabelle



6.4 Die 3. Normalform

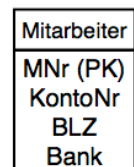
Definition 3. Normalform (3. NF)

Das Modell befindet sich in der 2. Normalform und es existieren keine (Nicht-Primärschlüssel)-Attribute, die untereinander abhängig sind.*

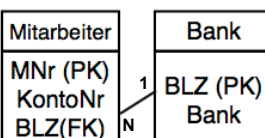
* Solche Abhängigkeiten werden auch als transitive Abhängigkeiten bezeichnet.

Beispiel zur 3. Normalform:

Problem 1: BLZ und Name des Geldinstituten sind redundante Informationen. Redundante Daten sind Auslöser für inkonsistente Daten.



Lösung: Die Attribute, die voneinander abhängig sind, werden in einer neuen Tabelle abgebildet (1-N Beziehung). Eins dieser Attribute wird Primärschlüssel.



Übrigens: Sollen mehrere Bankverbindungen möglich sein, ändert sich die Beziehung in eine N : M Beziehung und die KontoNr muss in die Beziehungstabelle.

Hinweise zur 3. NF:

In der Praxis ist es üblich PLZ und Ort in einer Tabelle zusammen zu lassen. Grund: zu einer PLZ können mehrere Orte gehören und umgekehrt.

Attribute, die aus anderen Attributen berechnet werden können, verletzen ebenfalls die 3. Normalform. Beispiel: Gesamtpreis. Es werden nur der Einzelpreis und die Anzahl gespeichert. Das berechnete Attribut Gesamtpreis ist zu entfernen.

7 Referenzielle Integrität

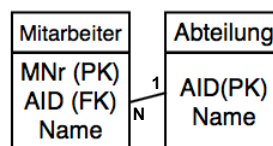
Definition Referenzielle Integrität:

Referenzielle Integrität bedeutet, dass zu jedem Fremdschlüssel-Eintrag in der Datenbank der entsprechende Primärschlüssel-Eintrag existieren muss. (Es gibt keine "verwaisten" Fremdschlüssel).

Mit der Referentiellen Integrität werden die Beziehung zwischen Datensätzen kontrolliert.

- Ein neuer Datensatz mit einem Fremdschlüssel kann nur dann in einer Tabelle eingefügt werden, wenn in der referenzierten Tabelle ein Datensatz mit entsprechendem Wert im Primärschlüssel existiert.
- Eine Datensatzlöschung oder Änderung des Schlüssels in einem Primär-Datensatz ist nur möglich, wenn zu diesem Datensatz keine abhängigen Fremdschlüssel-Datensätze in Beziehung stehen.

Beispiel zur Referenziellen Integrität



Aufgrund der 1-N Beziehung muss in der Tabelle Mitarbeiter die AID als Fremdschlüssel abgebildet werden.

Mitarbeiter		
MID(PK)	Name	AID (FK)
1	Müller	77
2	Meier	77
3	Schmidt	99

Abteilung	
Name	AID(PK)
Personal	77
Entwicklung	99

Wird in der Tabelle Abteilung der Eintrag AID = 99 gelöscht, ist ein Fremdschlüssel-Eintrag 99 in der Tabelle Mitarbeiter nicht mehr zuzuordnen. Fremdschlüssel-Einträge müssen immer zu einem existierenden Primärschlüssel verweisen!

In der Regel wird dies automatisch von der Datenbank überprüft. Das heißt, dass das Löschen nur erlaubt wird, wenn keine Fremdschlüssel-Einträge mehr in anderen Tabellen existieren. Auch das Anlegen wird überprüft. Es kann kein Mitarbeiter angelegt werden mit einer AID, die nicht in der Tabelle Abteilung existiert.