Datenbanksysteme

Imed Ghaouari

Einführung

Imed Ghaouari

Motivation

- Wir haben Daten, die gespeichert werden müssen.
- Wir könnten die Daten in Dateien wie z.B. Excel oder Word speichern.
- Und wir könnten sie in Ordnern hierarchisch ablegen.

Ziel

- Wir wollen die Daten verarbeiten, um daraus Informationen zu gewinnen und verknüpfen, um an neues Wissen zu kommen.

DATEN → INFORMATIONEN → WISSEN

Es geht bei Datenbanken um, die Wissensgewinnung.

Datenbanksysteme

- Wann sollten wir die Verwendung eines Datenbanksystems in Erwägung ziehen?

- Wenn die folgenden Punkte eine Rolle spielen:
  - Menge
  - Genauigkeit
  - keine - Redundanz
  - Mehrfachnutzung

Datenbanksystem

- Ist ein **System** zur Elektronischen **Datenverwaltung**.

Aufgabe eines **Datenbanksystems**

- Die Aufgabe der **DBS** ist es große Datenmengen effizient, widerspruchsfrei und dauerhaft zu speichern.
- Stellt benötigte Teilmengen in verschiedenen Darstellungsformen bereit.

Ein **Datenbanksystem** (**DBS**) besteht aus zwei Teilen:

- Aus der Menge der zur Verwalteten Daten DB
  - (**DatenBasis** / **DatenBank**)

- Aus der Verwaltungssoftware **DBMS**
  - (**DatenBankManagementSystem**)

DBS-Detail

Abbildung 1.1: Vereinfachte Datenbanksystemumgebung zur Darstellung der in Abschnitt 1.1 erklärten Konzepte und Fachbegriffe.

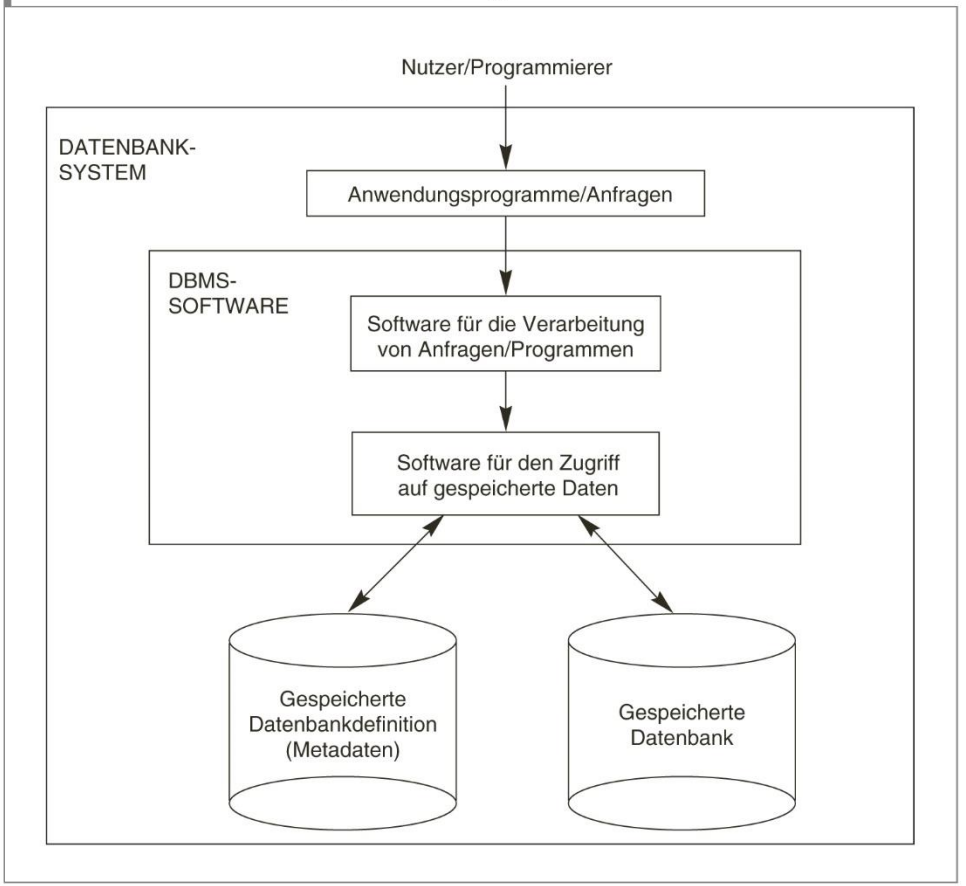


Abbildung 1.1 „Quelle: Grundlagen von Datenbanksystemen, Elmasri & Navathe, Pearson Deutschland GmbH, 3. Auflage 2009, S. 20“

Datenbanktypen

Imed Ghaouari

Datenbanktypen - 1

- Relationale Datenbanken:
  - Diese sind am weitesten verbreitet und organisieren Daten in Tabellen, die durch Beziehungen miteinander verbunden sind.

- Objektorientierte Datenbanken:
  - Diese speichern Daten in Form von Objekten, ähnlich wie in der objektorientierten Programmierung.

- Hierarchische Datenbanken:
  - Diese organisieren Daten in einer baumartigen Struktur, wobei jede Datenbankeinheit nur eine übergeordnete Einheit hat.

Datenbanktypen - 2

- Netzwerkdatenbanken:
  - Diese verwenden ein flexibleres Modell als hierarchische Datenbanken und erlauben komplexe Beziehungen zwischen Daten.

- NoSQL-Datenbanken:
  - Diese sind für große Mengen unstrukturierter Daten ausgelegt und bieten hohe Skalierbarkeit.

- Graphdatenbanken:
  - Diese speichern Daten in Form von Knoten und Kanten und sind ideal für die Darstellung und Abfrage von Netzwerken.

Relationale Datenbanken - 1

- Struktur:
  - Daten werden in Tabellen organisiert, die aus Zeilen und Spalten bestehen. Jede Zeile stellt einen Datensatz dar, und jede Spalte repräsentiert ein Attribut des Datensatzes.

- Beziehungen:
  - Verwenden Primär- und Fremdschlüssel, um Beziehungen zwischen Tabellen zu definieren.

Relationale Datenbanken - 2

- Vorteile:
  - Hohe Datenintegrität und Konsistenz, unterstützt komplexe Abfragen und Transaktionen.

- Nachteile:
  - Kann bei sehr großen Datenmengen und hohen Schreiblasten weniger performant sein.

- MySQL, PostgreSQL, Oracle, Microsoft SQL Server

Objektorientierte Datenbanken - 1

- Struktur:
  - Speichern Daten als Objekte, ähnlich wie in der objektorientierten Programmierung. Objekte können Methoden und Attribute enthalten.

- Beziehungen:
  - Unterstützen Vererbung, Polymorphismus und andere objektorientierte Konzepte.

Objektorientierte Datenbanken - 2

- Vorteile:
  - Natürliche Integration mit objektorientierten Programmiersprachen, gut für komplexe Datenstrukturen.
- Nachteile:
  - Weniger verbreitet und unterstützt als relationale Datenbanken, kann komplexer zu verwalten sein.
- db4o, ObjectDB

Hierarchische Datenbanken - 1

- Struktur:
  - Organisieren Daten in einer baumartigen Struktur mit einer strengen Eltern-Kind-Beziehung.

- Beziehungen:
  - Jede Einheit hat nur eine übergeordnete Einheit, was die Navigation vereinfacht.

Hierarchische Datenbanken - 2

- Vorteile:
  - Sehr schnelle Abfragen für hierarchisch strukturierte Daten.
- Nachteile:
  - Unflexibel bei Änderungen der Datenstruktur, schwer zu skalieren.
- IBM's Information Management System (IMS).

Netzwerkdatenbanken - 1

- Struktur:
  - Verwenden ein flexibles Modell mit komplexen Beziehungen zwischen Daten.

- Beziehungen:
  - Unterstützen viele-zu-viele-Beziehungen, was eine flexiblere Datenmodellierung ermöglicht.

Netzwerkdatenbanken Datenbanken - 2

- Vorteile:
  - Flexibel und leistungsfähig bei komplexen Datenbeziehungen.
- Nachteile:
  - Komplexer zu verwalten und zu navigieren als hierarchische oder relationale Datenbanken.
- Integrated Data Store (IDS)

NoSQL-Datenbanken - 1

- Struktur:
  - Speichern Daten in verschiedenen Formaten wie Dokumenten, Key-Value-Paaren, Spalten oder Graphen.

- Beziehungen:
  - Flexibler als relationale Datenbanken, oft ohne festes Schema.

NoSQL-Datenbanken - 2

- Vorteile:
  - Hohe Skalierbarkeit und Leistung, besonders für große Mengen unstrukturierter Daten.

- Nachteile:
  - Weniger Unterstützung für komplexe Abfragen und Transaktionen, inkonsistente Datenmodelle.

- MongoDB, Cassandra, Redis

Graphdatenbanken - 1

- Struktur:
  - Speichern Daten in Form von Knoten und Kanten, die Netzwerke von Beziehungen darstellen.

- Beziehungen:
  - Optimiert für die Darstellung und Abfrage von Netzwerken und Beziehungen.

Graphdatenbanken - 1

- Vorteile:
  - Sehr leistungsfähig für Abfragen, die komplexe Beziehungen betreffen, wie z.B. soziale Netzwerke.

- Nachteile:
  - Kann bei einfachen, tabellenbasierten Datenmodellen weniger effizient sein.

- Neo4j

Skalierung – Arten

- Vertikale Skalierung:
  - Erhöhung der Hardware-Ressourcen eines einzelnen Servers. Ist durch die physikalischen Grenzen eines einzelnen Servers begrenzt.

- Horizontale Skalierung:
  - Verteilung der Daten auf mehrere Server (Sharding). Dies erfordert komplexe Datenverteilungs- und Synchronisierungsmechanismen, bietet aber eine höhere Skalierbarkeit.

Skalierung – Arten

- Die Skalierung von Datenbanken kann auf verschiedene Arten umgesetzt werden, je nach Typ der Datenbank und den spezifischen Anforderungen.

- Die Wahl der Skalierungsmethode hängt von den spezifischen Anforderungen und der Architektur der Anwendung ab. Horizontale Skalierung bietet oft eine bessere Langzeitlösung für große Datenmengen und hohe Lasten, während vertikale Skalierung einfacher umzusetzen ist, aber durch die physikalischen Grenzen eines einzelnen Servers begrenzt ist.

Skalierung - Relationale Datenbanken

- Vertikale Skalierung:
  - Erhöhung der Hardware-Ressourcen eines einzelnen Servers.

- Horizontale Skalierung:
  - Verteilung der Daten auf mehrere Server (Sharding). Dies erfordert komplexe Datenverteilungs- und Synchronisierungsmechanismen, bietet aber eine höhere Skalierbarkeit.

Skalierung - Objektorientierte Datenbanken

- Vertikale Skalierung:
  - Ähnlich wie bei relationalen Datenbanken, durch Aufrüstung der Hardware-Ressourcen eines einzelnen Servers.

- Horizontale Skalierung:
  - Komplexer aufgrund der Objektbeziehungen, aber möglich durch verteilte Objekt-Caching-Mechanismen und Replikation.

Skalierung – Netzwerkdatenbanken

- Vertikale Skalierung:
  - Aufrüstung der Hardware-Ressourcen eines einzelnen Servers..

- Horizontale Skalierung:
  - Flexibler als hierarchische Datenbanken, da viele-zu-viele-Beziehungen unterstützt werden. Dies kann durch verteilte Datenbankarchitekturen und Replikation erreicht werden.

Skalierung - NoSQL-Datenbanken

- Vertikale Skalierung:
  - Erhöhung der Ressourcen eines einzelnen Servers, oft als kurzfristige Lösung verwendet.

- Horizontale Skalierung:
  - Hauptvorteil von NoSQL-Datenbanken. Daten werden auf mehrere Server verteilt (Sharding), was eine hohe Skalierbarkeit und Leistung ermöglicht.

Skalierung – Graphdatenbanken

- Vertikale Skalierung:
  - Aufrüstung der Hardware-Ressourcen eines einzelnen Servers.

- Horizontale Skalierung:
  - Komplexer aufgrund der Knoten- und Kantenstruktur. Kann durch verteilte Graphdatenbankarchitekturen und Replikation erreicht werden. Neo4j bietet beispielsweise Cluster- und Sharding-Optionen.

Partitionierung - 1

- Definition:
  - Partitionierung teilt eine große Datenbanktabelle in kleinere, logisch zusammenhängende Teile, die Partitionen genannt werden. Diese Partitionen werden innerhalb derselben Datenbankinstanz gespeichert.

- Arten der Partitionierung:
  - Horizontale Partitionierung
  - Vertikale Partitionierung

Partitionierung - 2

- Horizontale Partitionierung:
  - Teilt die Tabelle nach Zeilen auf. Jede Partition enthält eine Teilmenge der Zeilen der ursprünglichen Tabelle.

- Vertikale Partitionierung:
  - Teilt die Tabelle nach Spalten auf. Jede Partition enthält eine Teilmenge der Spalten der ursprünglichen Tabelle.

Partitionierung - 3

- Vorteile:
  - Verbessert die Leistung, indem Abfragen nur auf einer Partition ausgeführt werden müssen. Erleichtert die Verwaltung und Wartung großer Tabellen.

- Nachteile:
  - Kann komplex sein, wenn Abfragen Daten aus mehreren Partitionen benötigen.

…

- …
- …

Sharding - 1

- Definition:
  - Sharding ist eine Art der horizontalen Partitionierung, bei der eine große Datenbank in kleinere, unabhängige Teile, sogenannte Shards, aufgeteilt wird. Jeder Shard wird auf einem separaten Datenbankserver gespeichert.

- Arten des Shardings:
  - Horizontales Sharding
  - Vertikales Sharding

Sharding - 2

- Horizontales Sharding:
  - Teilt die Datenbank nach Zeilen auf, ähnlich wie bei der horizontalen Partitionierung. Jeder Shard enthält eine Teilmenge der Zeilen der ursprünglichen Datenbank.

- Vertikales Sharding:
  - Teilt die Datenbank nach Spalten auf, wobei jeder Shard eine Teilmenge der Spalten enthält.

Sharding - 3

- Vorteile:
  - Erhöht die Skalierbarkeit und Verfügbarkeit, da jeder Shard unabhängig auf einem separaten Server läuft. Reduziert die Last auf einzelnen Servern und verbessert die Leistung.

- Nachteile:
  - Erfordert eine komplexe Verwaltung und Synchronisation der Shards. Kann zu Dateninkonsistenzen führen, wenn nicht richtig implementiert.

Partitionierung vs Sharding

- Partitionierung:
  - Innerhalb derselben Datenbankinstanz, einfacher zu verwalten, aber begrenzte Skalierbarkeit.

- Sharding:
  - Über mehrere Datenbankserver verteilt, höhere Skalierbarkeit, aber komplexere Verwaltung.

- Beide Techniken haben ihre eigenen Anwendungsfälle und werden je nach den spezifischen Anforderungen und der Architektur der Anwendung ausgewählt. Möchtest

RDBS

Imed Ghaouari

Relationale Datenbank Management System (RDBMS)

Name	Firma	Released	Admin Programm	Lizenz	Kostenlos
Oracle	Oracle	1979	Oracle SQL Developer	Comm.	Express
DB2	IBM	1983	IBM Studio	Comm.	Express-C
SQL Server	Microsoft	1989	SQL Server Manag. Studio	Comm.	Express
MySQL	Oracle	1994	MySQL Workbench	Open.	Community

SQL Server EXPRESS 2012 - Installationsvorrausetzung

Microsoft	Betriebssystem	Architektur	RAM	HD
Mindestens	Win. Vista (SP 2); Win. Server 2008 (SP 2);	32-Bit-Systeme, (Intel) 1-GHz-CPU; 64-Bit-Systeme, (Intel) 1,4-GHz-CPU;	512 MB	2,2 GB
Empfohlen	Windows 7; Windows Server 2008 R2;	2-GHz-CPU	2 GB	2,2 GB

Komponenten

Abbildung 2.3: Typische Komponenten eines DBMS; die gepunkteten Linien stellen Zugriffe dar, die unter der Kontrolle des Storage Manager ausgeführt werden.

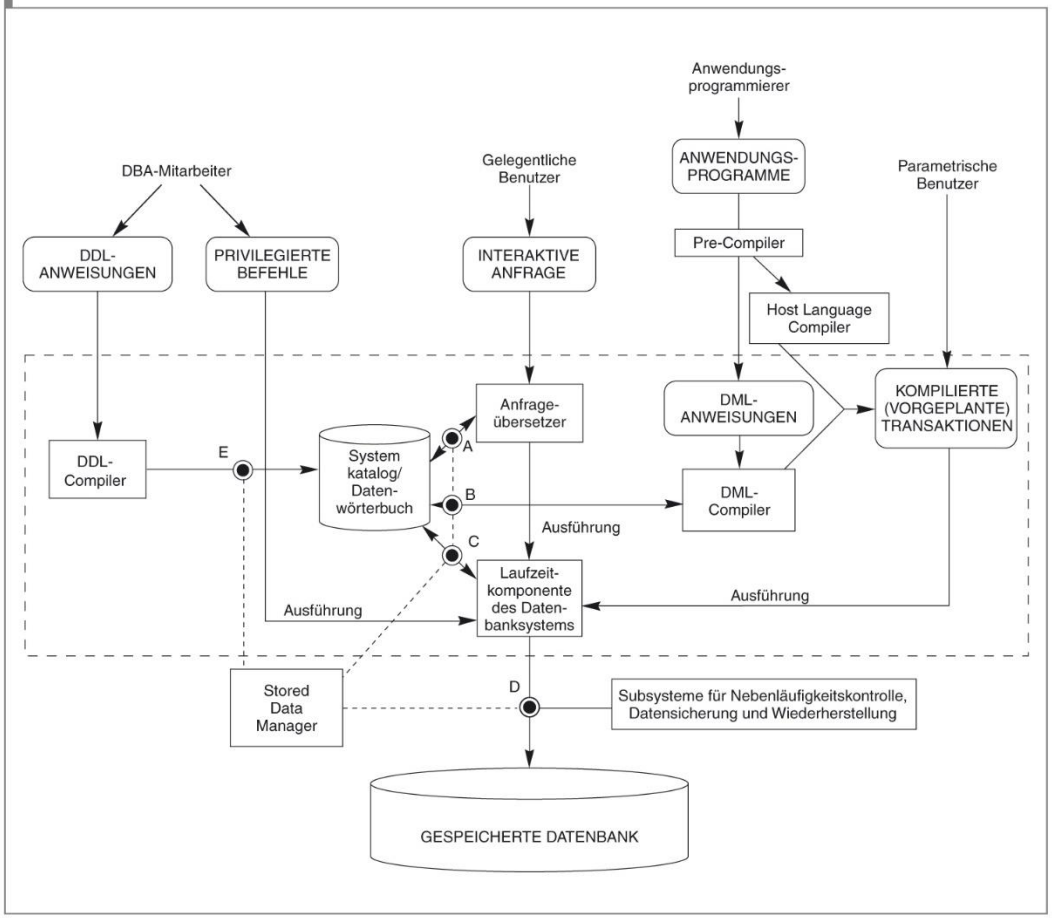


Abbildung 2.3 „Quelle: Grundlagen von Datenbanksystemen, Elmasri & Navathe, Pearson Deutschland GmbH, 3. Auflage 2009, S. 50“

Datenbanktheorie

Imed Ghaouari

3-Schichten

Abbildung 2.2: Darstellung der Drei-Schichten-Architektur.

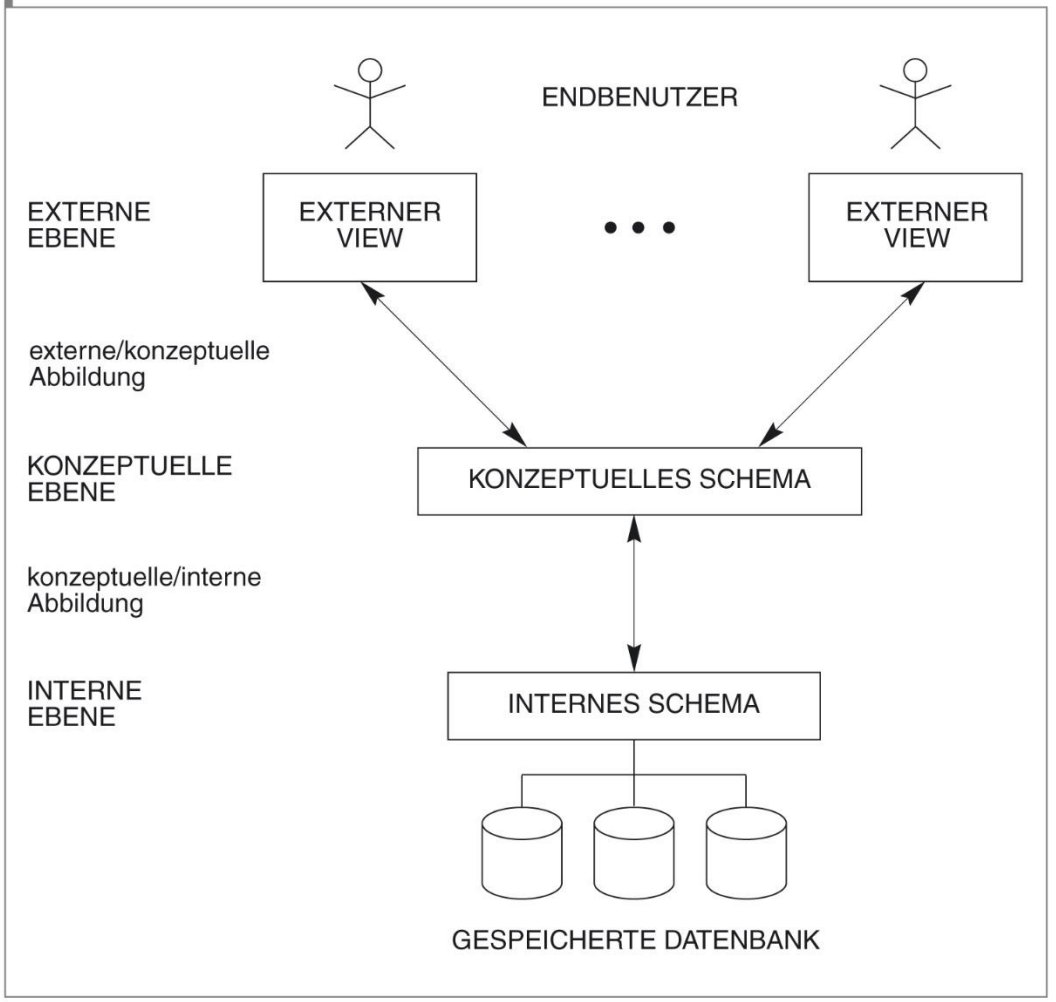


Abbildung 2.2 „Quelle: Grundlagen von Datenbanksystemen, Elmasri & Navathe, Pearson Deutschland GmbH, 3. Auflage 2009, S. 44"

Modellieren

1. Beim Modellieren vereinfachen wir die Wirklichkeit in dem wir aus der Realität einen kleinen Ausschnitt nehmen.

2. Wir nehmen nur Objekte auf, die zum Vorhaben passen.

3. Wir Gruppieren gleiche Elemente.

4. Wir beschreiben die Eigenschaften der Objekte durch Ihre Attribute.

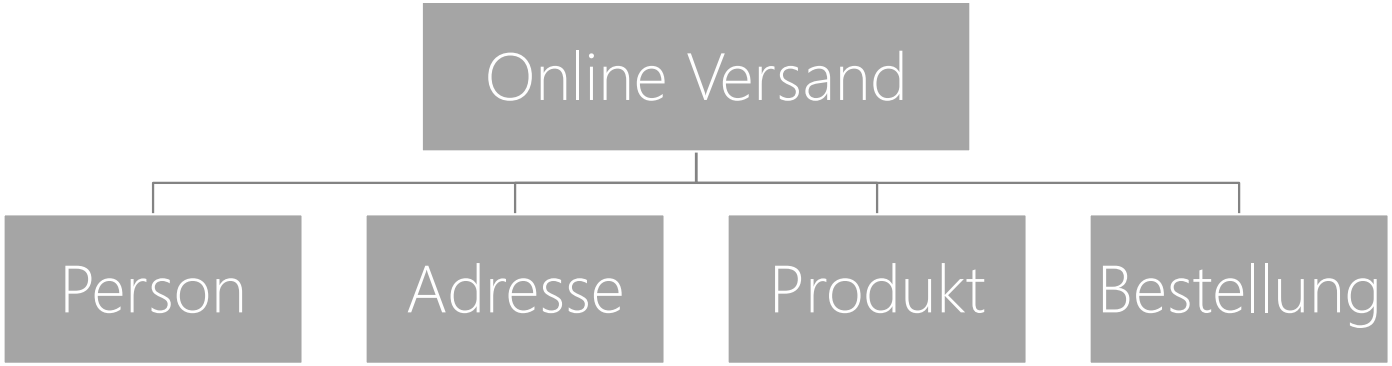
5. Wir definieren die Zusammenhänge der Objekte.

Kundenauftrag Beispiel

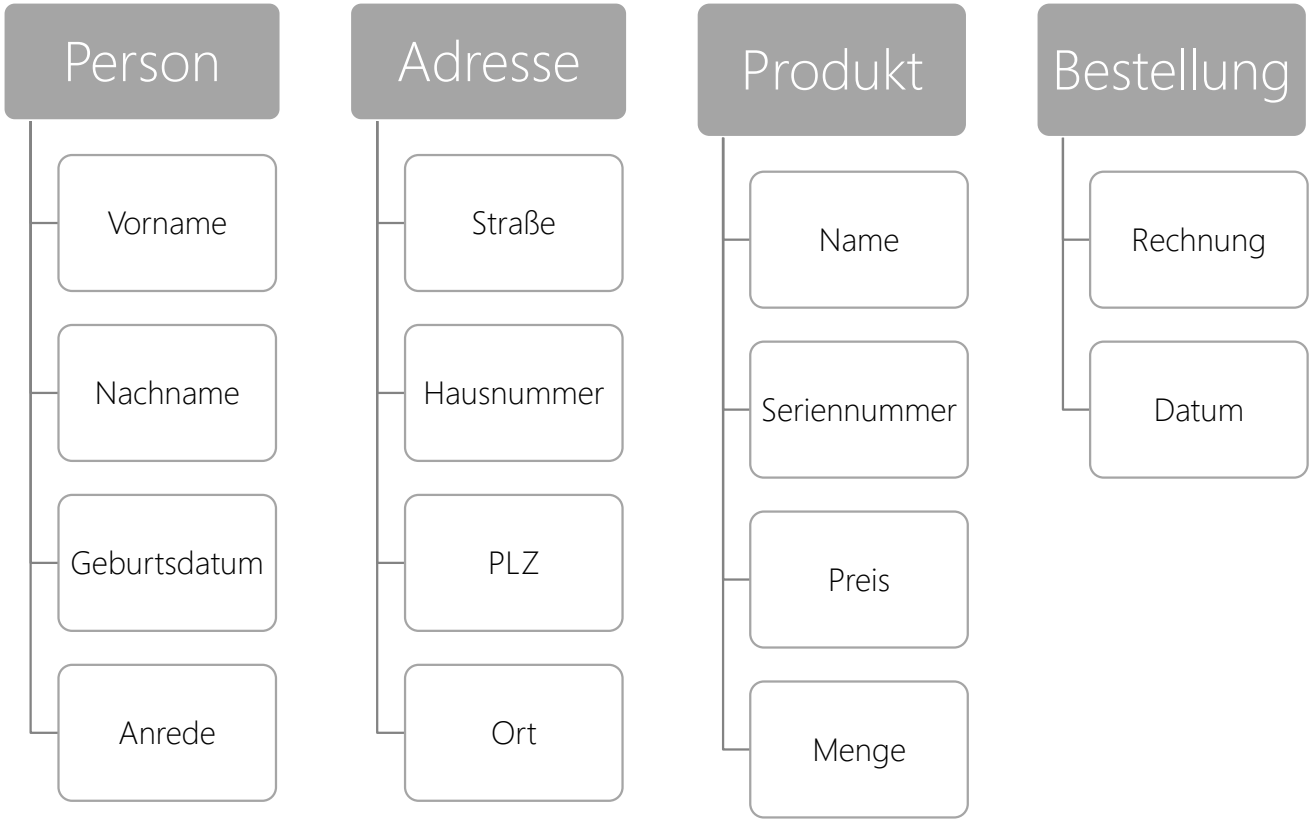
1. Wir nehmen einen kleinen Ausschnitt aus der Realität:
   - Online Versand / Computer Hardware

2. Wie nehmen nur Objekte auf, die zu zum Vorhaben passen:
   - Person, Vorname, Nachname, Anrede, Geburtsdatum, Geschlecht
   - Adresse, Straße, Hausnummer, Postleitzahl, Ort
   - Produkt, Artikelnummer, Bezeichnung, Preis, Anzahl
   - Bestellung, Rechnung, Datum

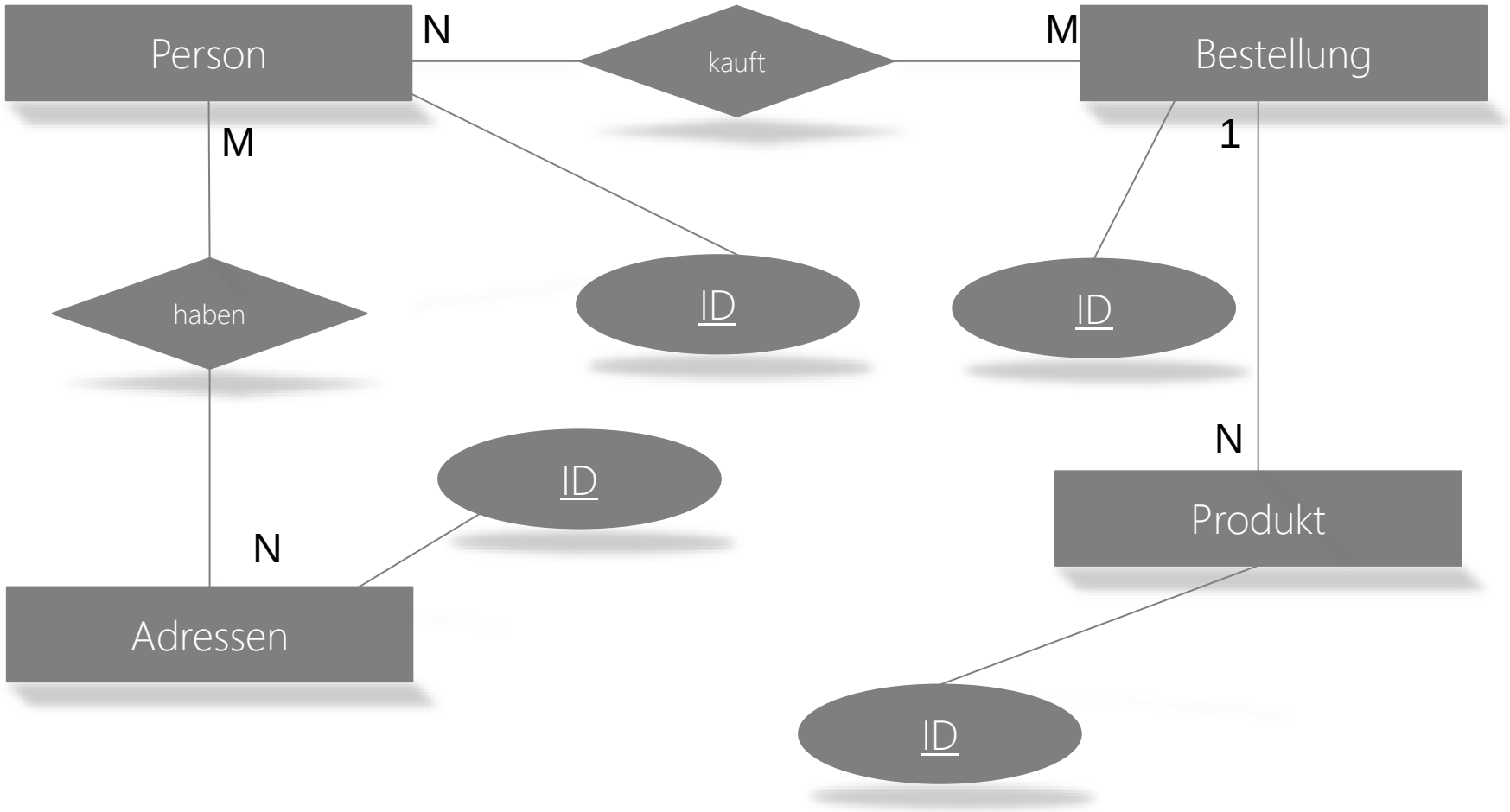
3. Wir Gruppieren gleiche Elemente:



4. Wir beschreiben die Eigenschaften der Objekte durch Ihre Attribute:

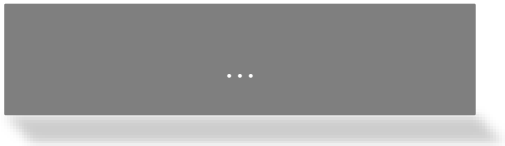


5. Wir definieren die Zusammenhänge der Objekte (Beziehung):

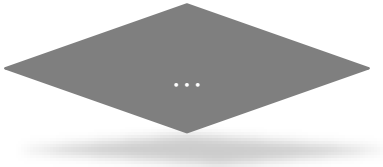


Enity-Relationship-Diagram

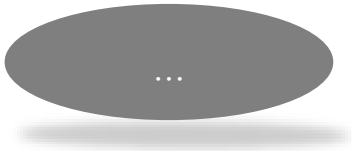
- Entity



- Relationship



- Attribute



Verb-Substantiv-Methode

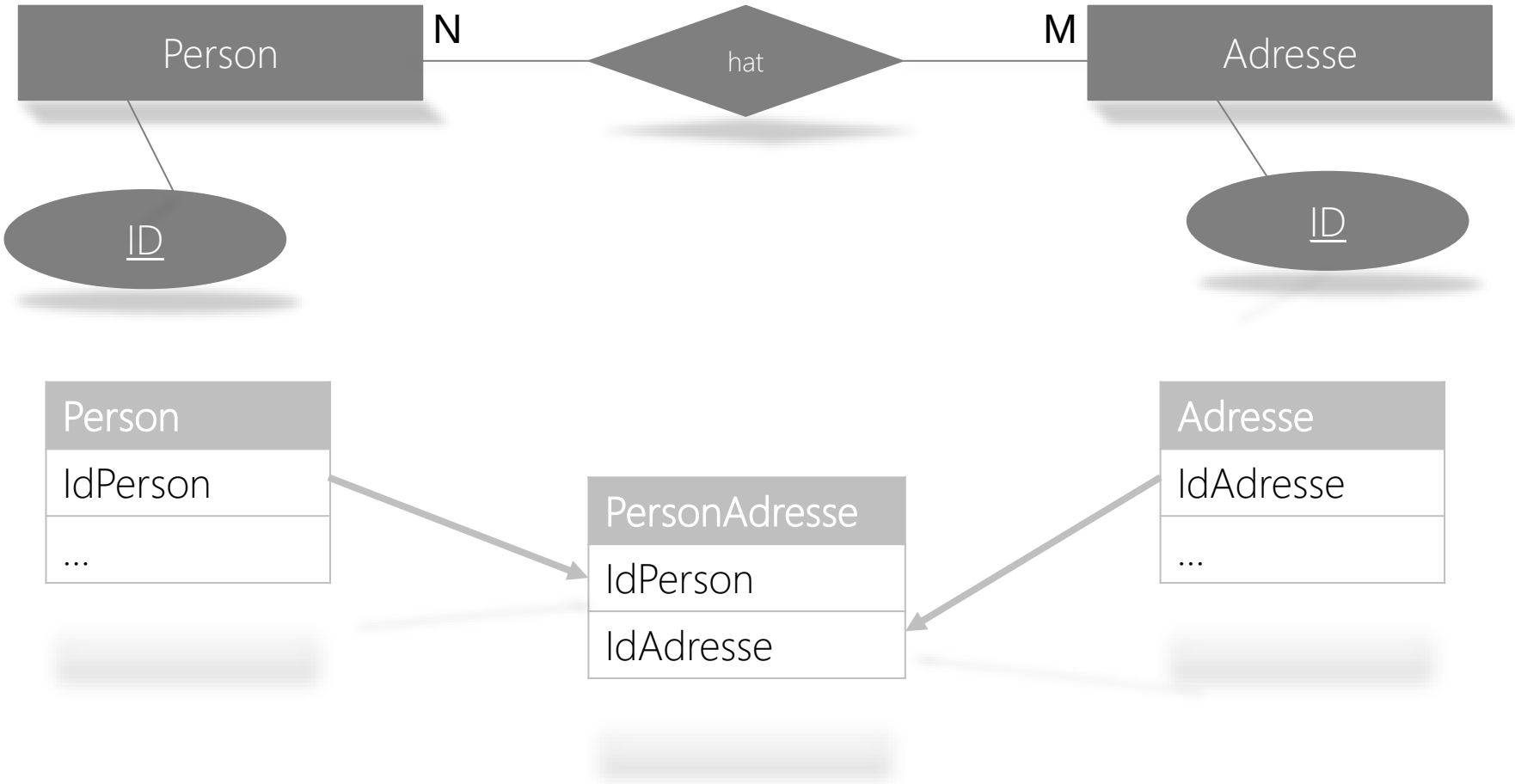
- Kunden haben Adressen. Ihre Adresse besteht aus der Straße, Hausnummer und eine Postleitzahl. Die Kunden haben einen Namen eine Kundenummer und eine E-Mail - Adresse.

- Entitätstypen: Kunden, Adressen
- Attribute Kunden: Namen, Kundennummer, E-Mail
- Attribute Adressen: Straße, Hausnummer, Postleitzahl
- Beziehung: haben
- Kardinalitäten: Kunde(n), Adress(en) → Plural

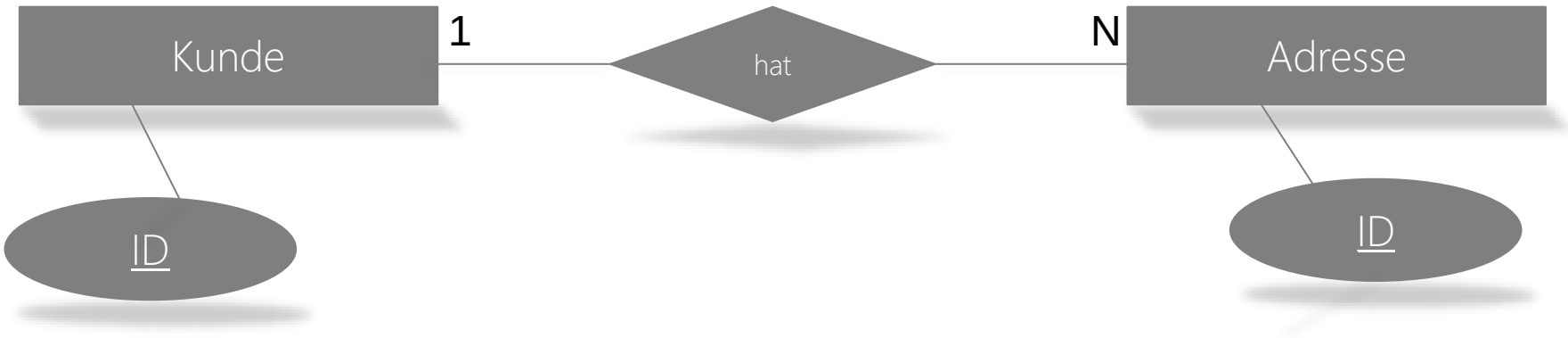
Multiplizität / Kardinalität

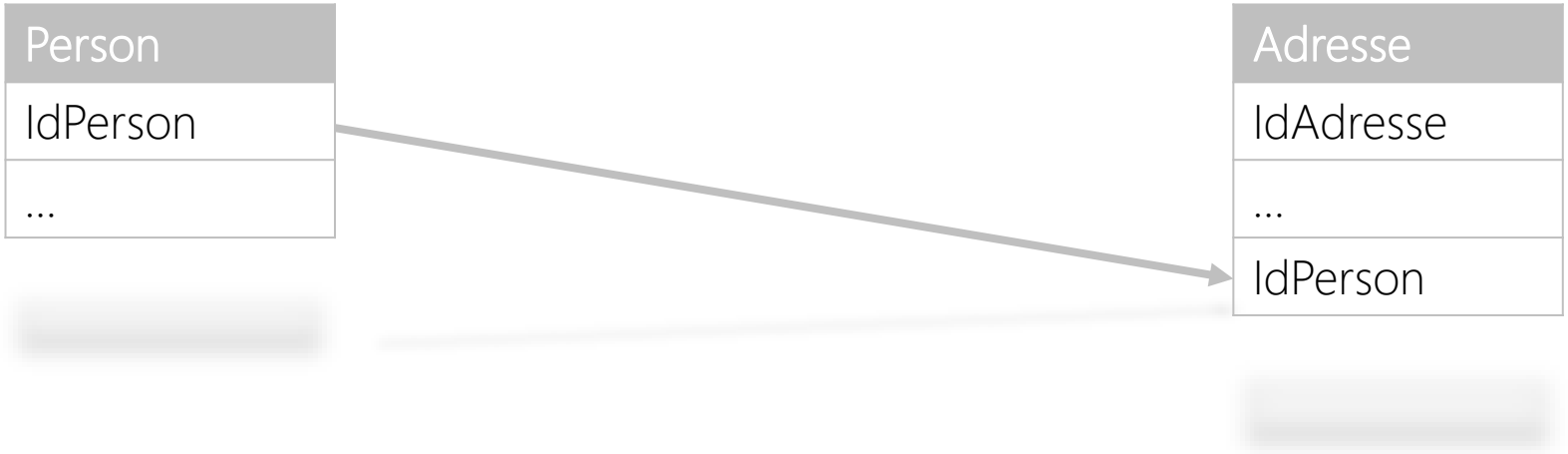
- 1 zu 1
- Viele zu 1
- 1 zu Viele
- Viele zu Viele

Viele Personen haben viele Adressen:

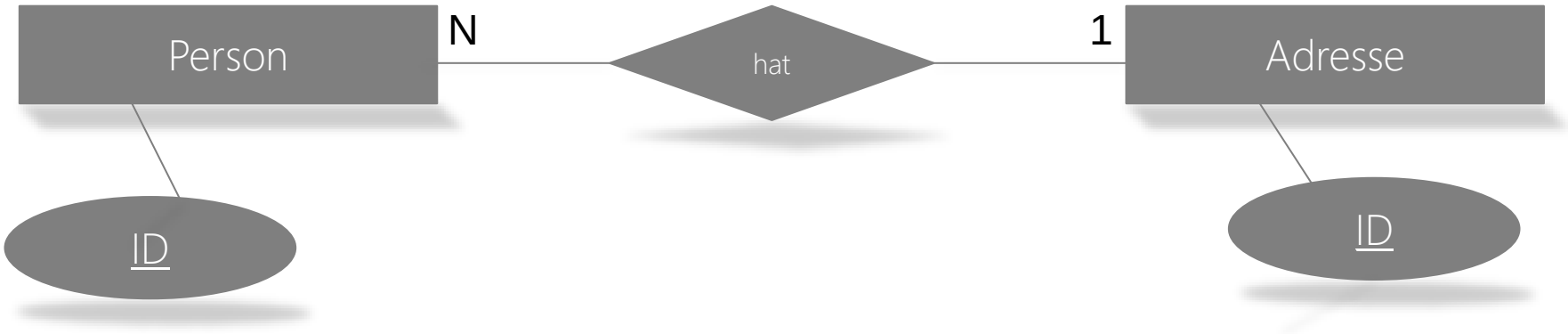


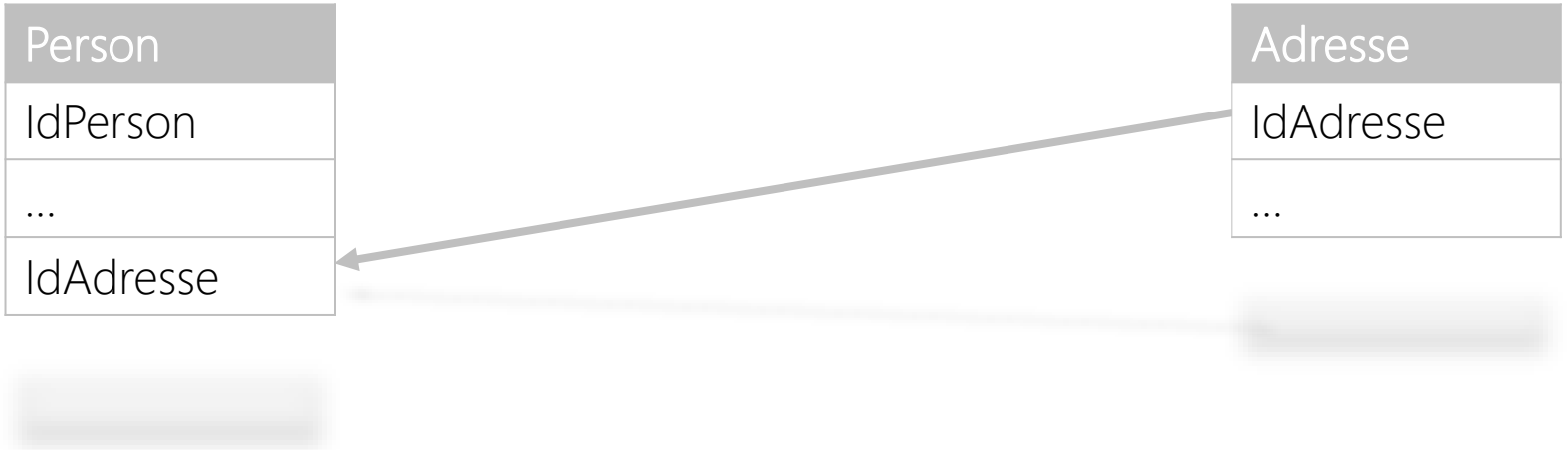
Eine Person hat viele Adressen



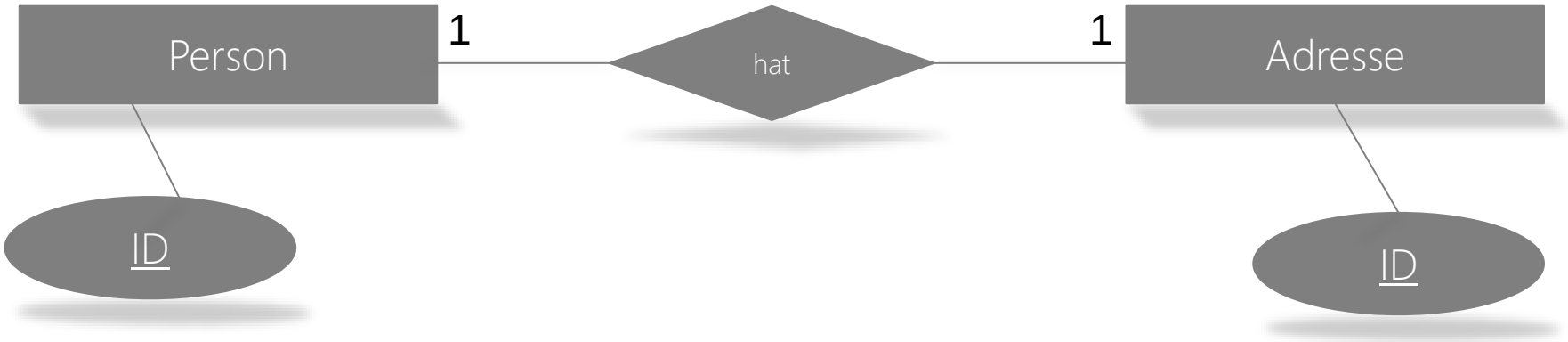


Viele Personen haben eine Adresse:



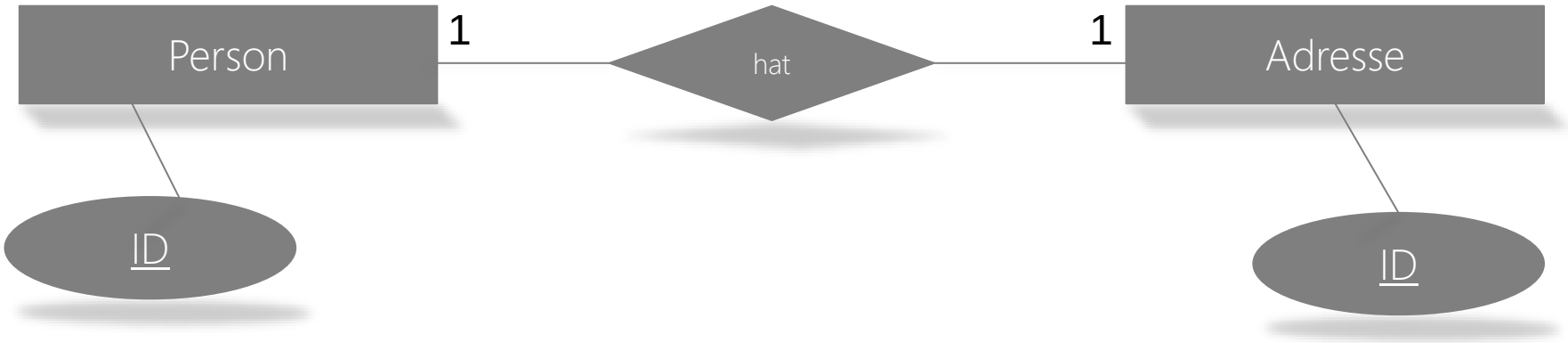


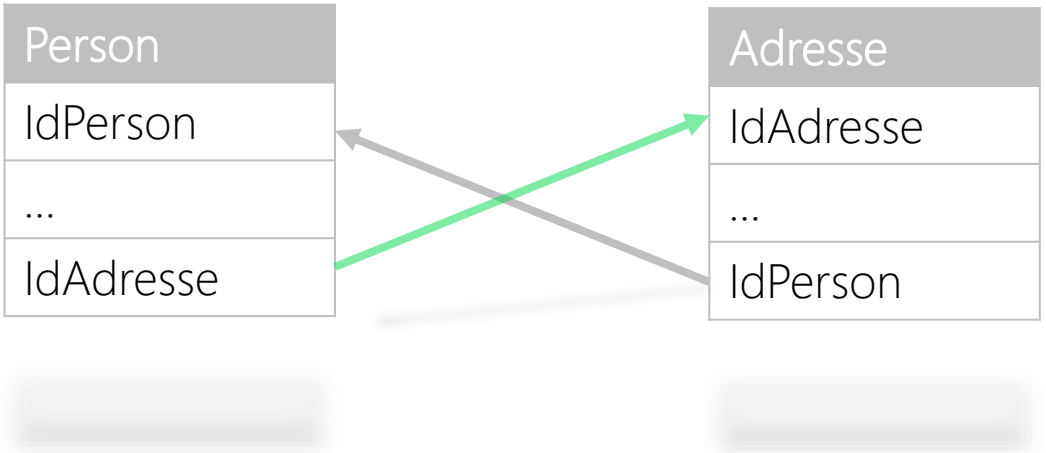
Eine Person hat eine Adresse (nur eine Tabelle):



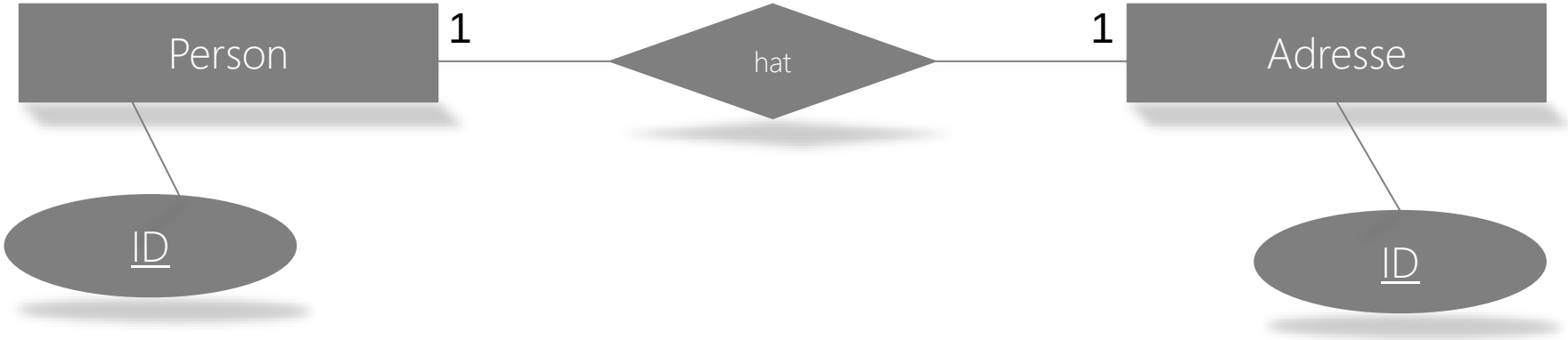
Person
IdPerson
…

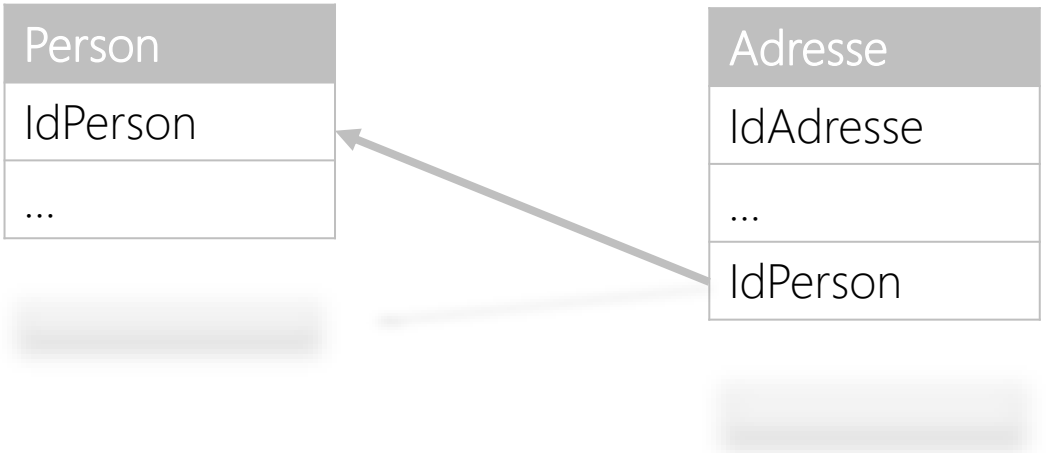
Eine Person hat eine Adresse (ID's in beiden Tabellen):





Eine Person hat eine Adresse (Unique–Key):





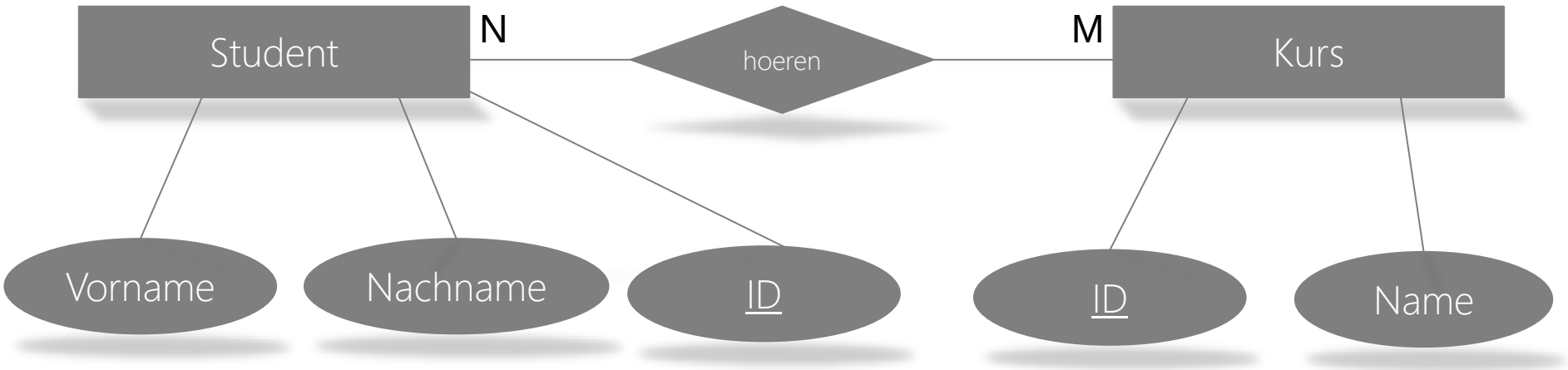
...

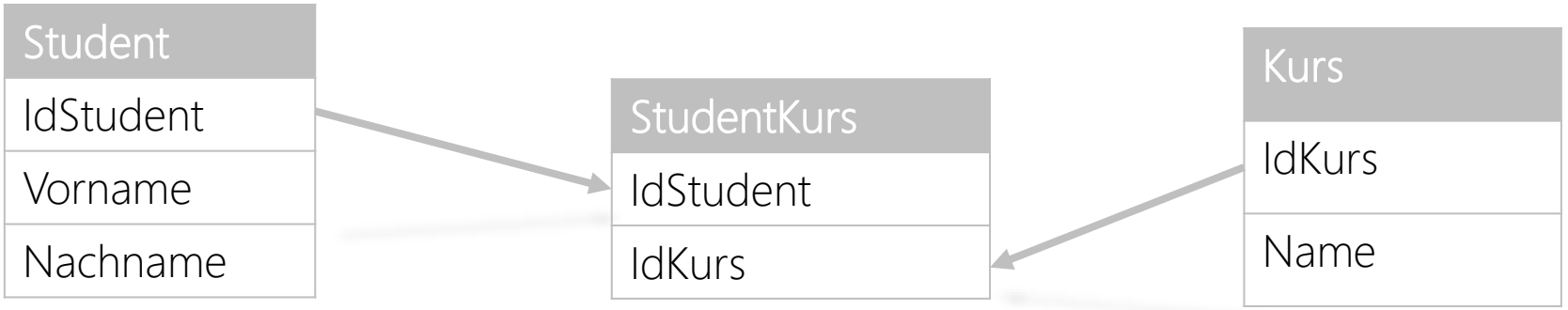

Mitarbeiter
IDMitarbeiter
Vorname
Nachname
Notebook
IDNotebook
Hersteller
Seriennummer
IDMitarbeiter

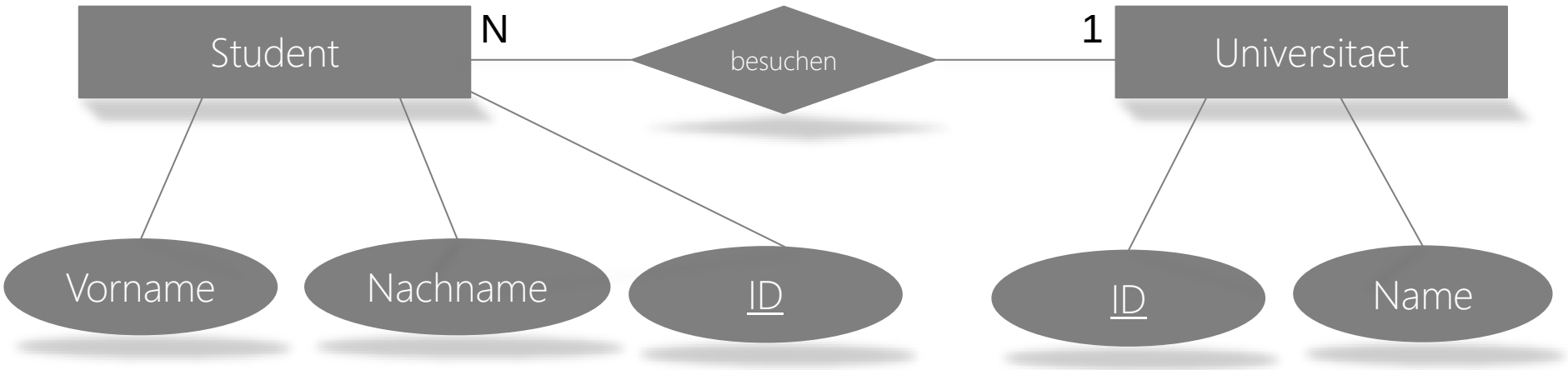

Mitarbeiter
IDMitarbeiter
Vorname
Nachname
IDNotebook
Notebook
IDNotebook
Hersteller
Seriennummer
IDMitarbeiter

Tabellen Design

- Entitätstypen → Tabellen
- Attribute → Spalten
- Beziehungen → Verbindung über Schlüssel
  - N-1 → **ID** der **1-Seite** in **N-Seite** einfügen
  - N-M → **ID** der **N-Seite** & **ID** der **M-Seite** in eine **neue Tabelle** einfügen
  - 1-1 → **ID** der **N-Seite** & **ID** der **M-Seite** in die **jeweilige andere Tabelle** einfügen

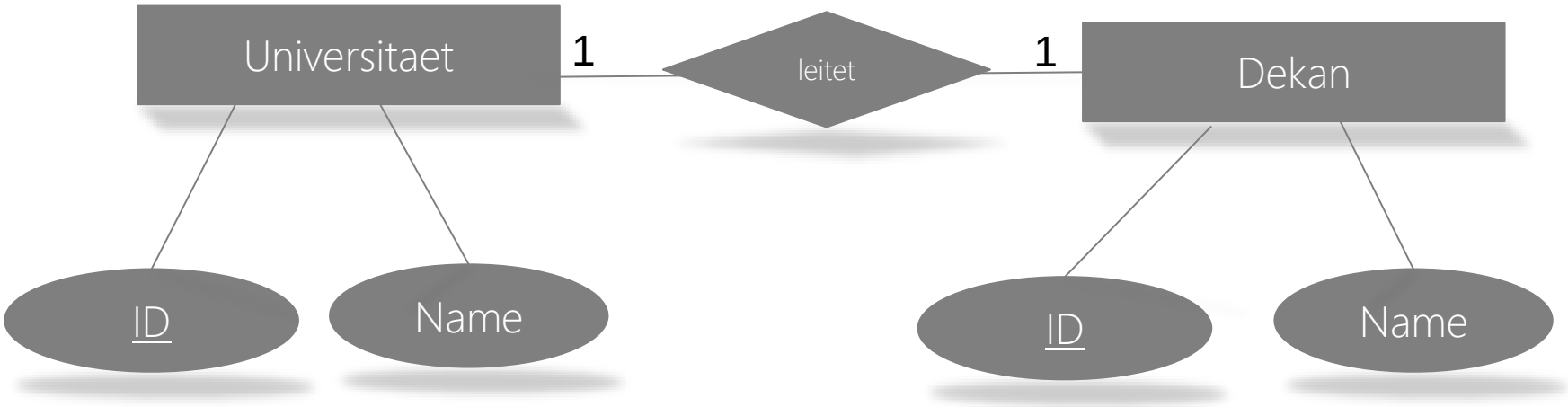

Student
N
hoeren
M
Kurs
Vorname
Nachname
ID
ID
Name

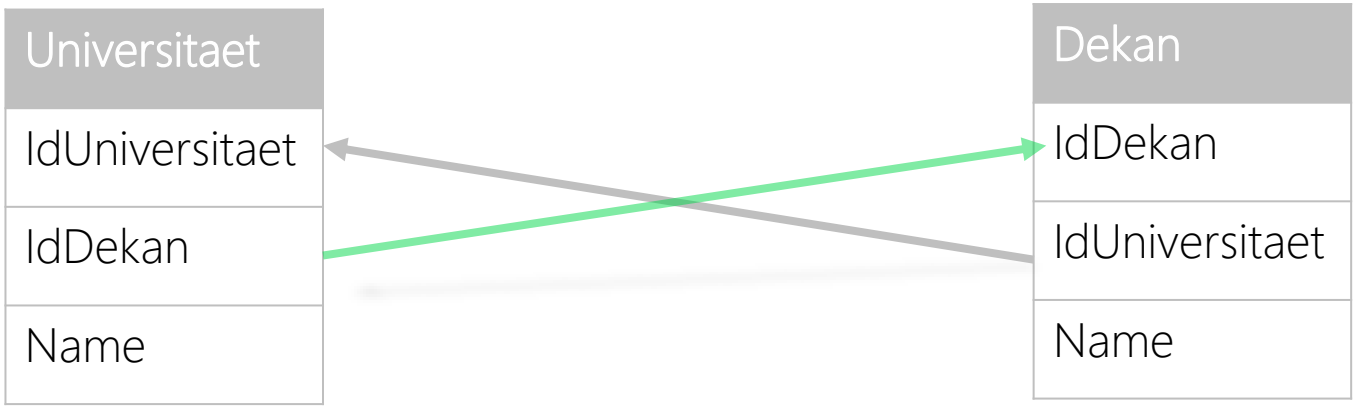

Student
IdStudent
Vorname
Nachname
StudentKurs
IdStudent
IdKurs
Kurs
IdKurs
Name

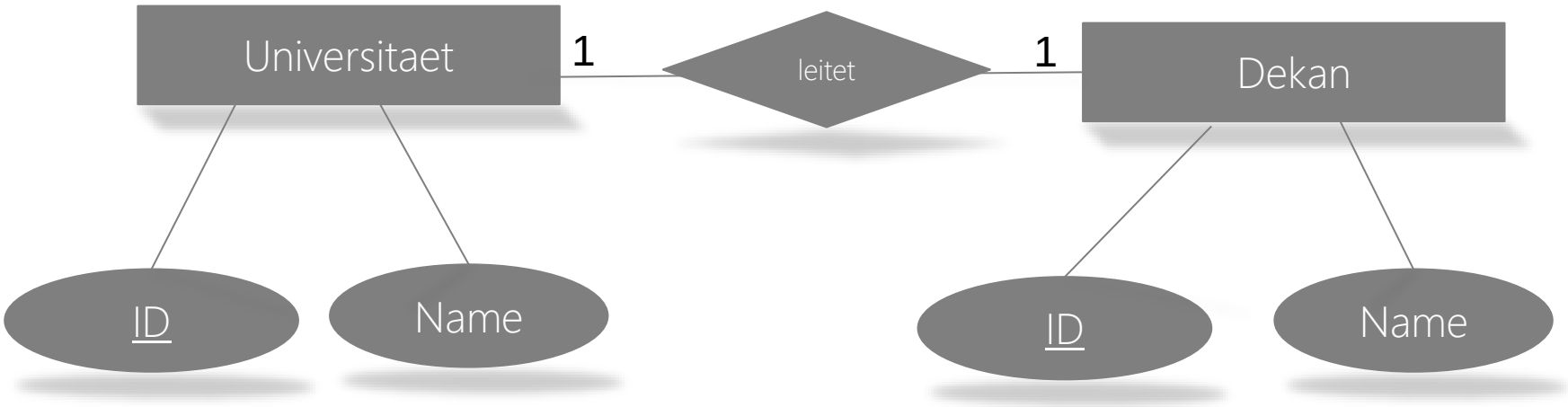


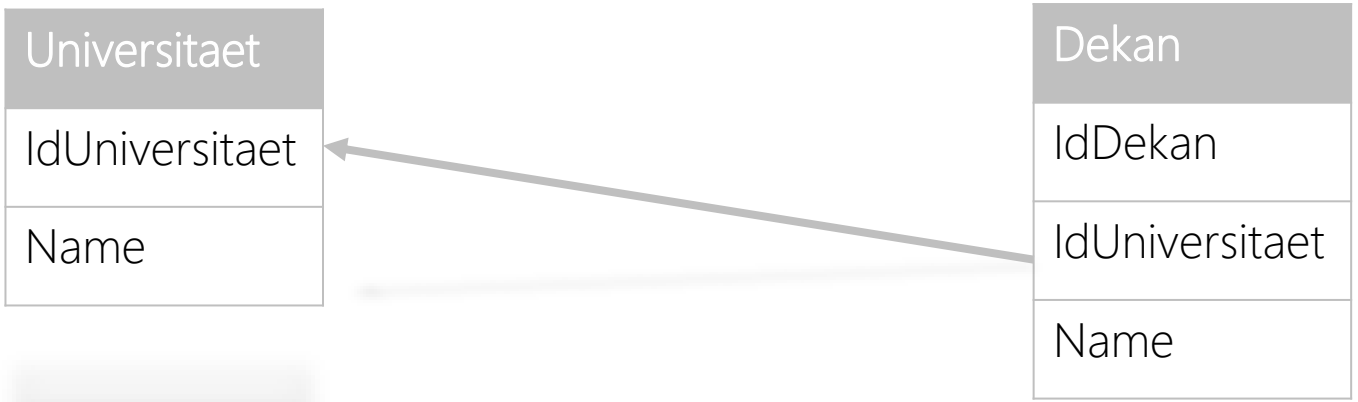
Student
IdStudent
IdUniversitaet
Vorname
Nachname

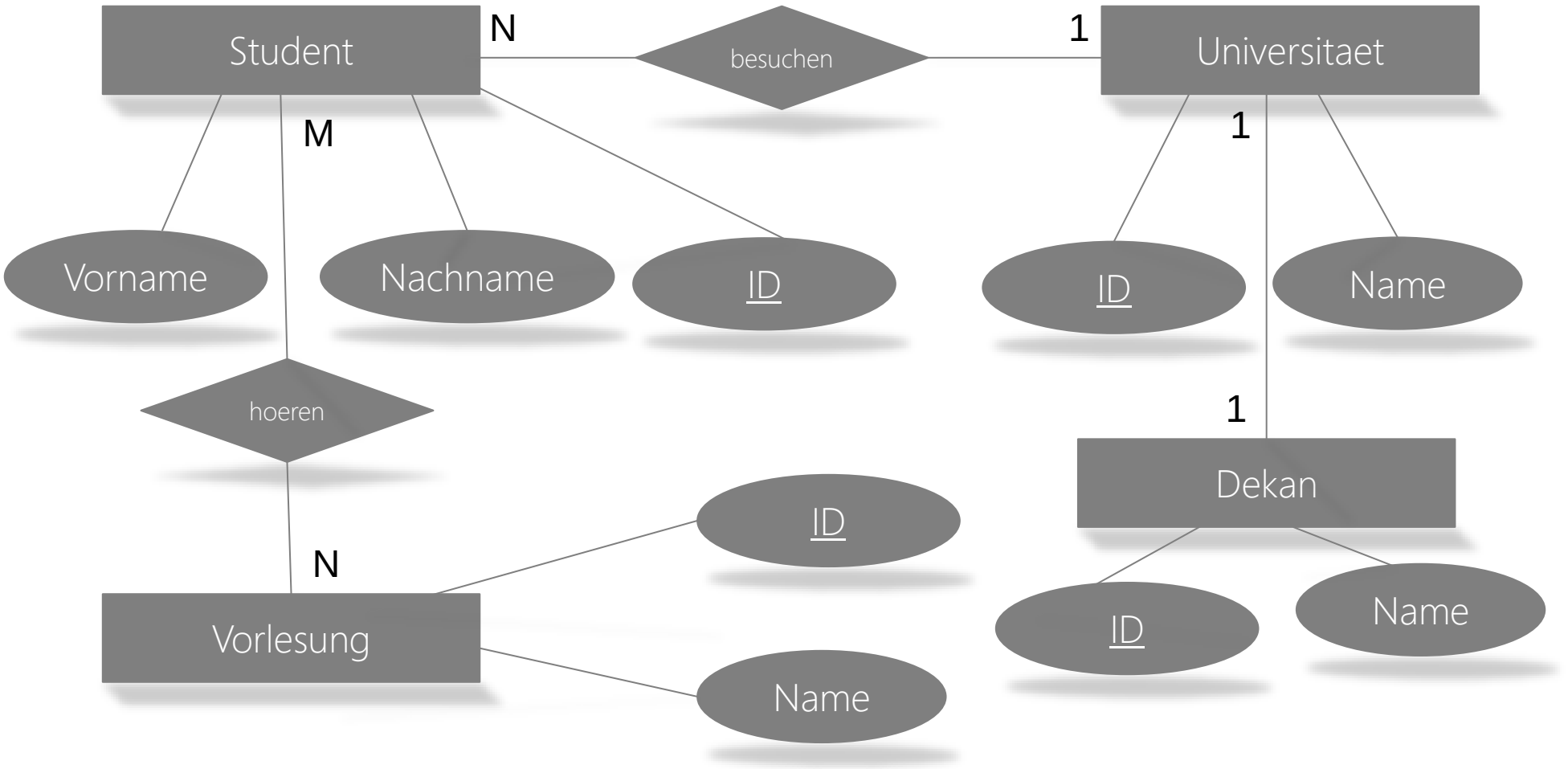
Universitaet
IdUniversitaet
Name

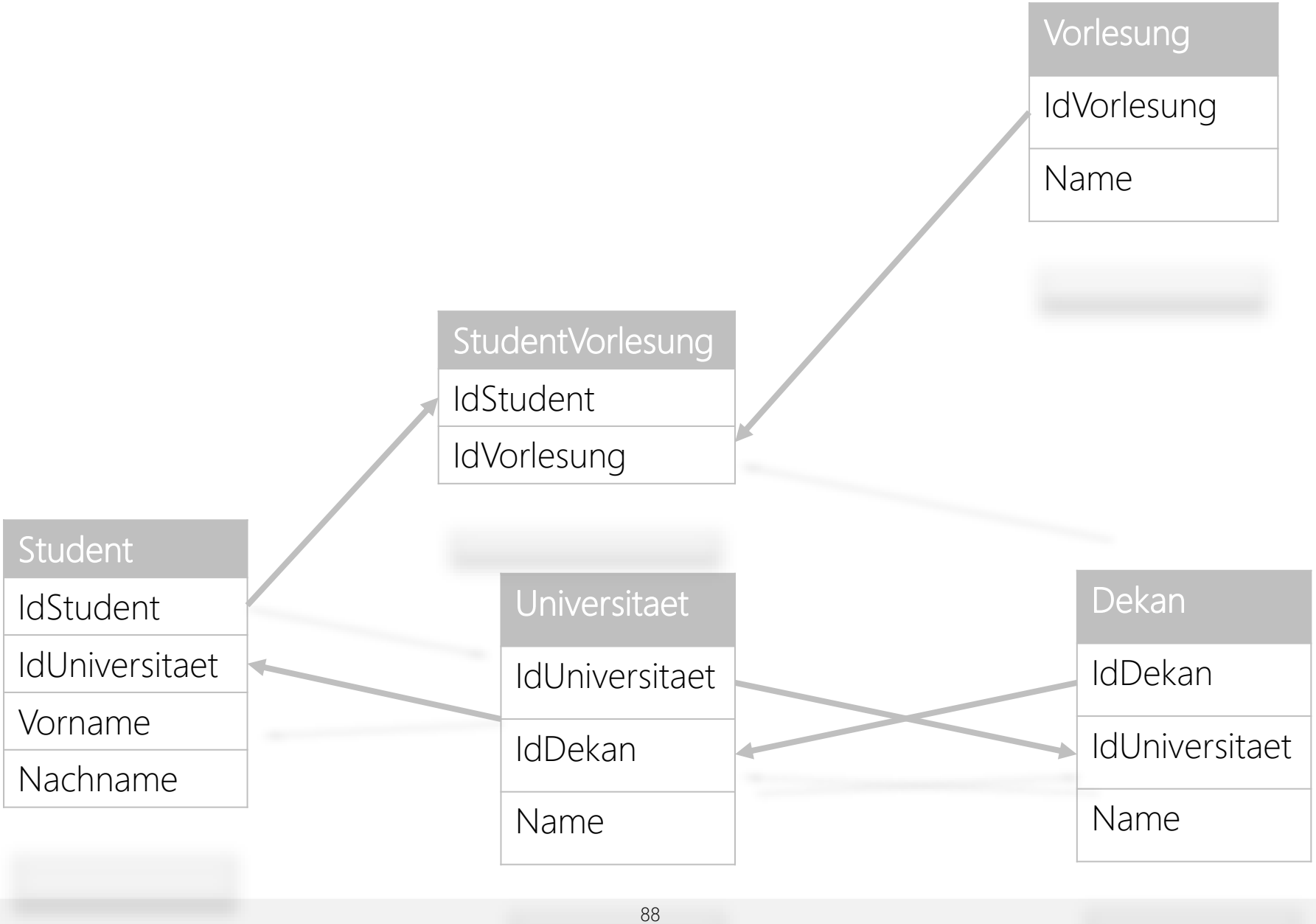

Universitaet
1
leitet
1
Dekan
ID
Name
ID
Name


Universitaet
IdUniversitaet
IdDekan
Name
Dekan
IdDekan
IdUniversitaet
Name


Universitaet
1
leitet
1
Dekan
ID
Name
ID
Name


Universitaet
IdUniversitaet
Name
Dekan
IdDekan
IdUniversitaet
Name


Student
N
besuchen
1
Universitaet
M
1
Vorname
Nachname
ID
ID
Name
hoeren
1
Dekan
ID
N
Vorlesung
Name
ID
Name


Vorlesung
IdVorlesung
Name
StudentVorlesung
IdStudent
IdVorlesung
Student
IdStudent
IdUniversitaet
Vorname
Nachname
Universitaet
IdUniversitaet
IdDekan
Name
Dekan
IdDekan
IdUniversitaet
Name

Schema

- Ist die Formale Beschreibung der DB.
- Tabellen, Spalten, Schlüssel, Beziehungen

- Es geht darum Regeln, Einschränkungen und Strukturen für die Daten fest zulegen.
- Das bedarf einer sehr guten Planung, da man von Anfang an darauf aus ist, alles richtig zu machen.

Constraints (Einschränkungen)

- Regeln, die auf Datenbank/Tabellen-Ebene erzwungen werden.
- Sie prüfen DML-Anweisungen bevor Sie angewendet werden.
- Die Regeln, können nicht umgegangen werden.
- Sie können nur ganz entfernt werden.

- Constraints - Typen:
  - Primary Key / Unique Key
  - Foreign Key
  - Check
  - Not Null

Primary Key

- Erzwingt die Eindeutigkeit.
- Die als Primary Key definierte Spalte ist automatisch nicht NULL.
- Ein Index durch schnelles finden wird erstellt.
- Jede Tabelle sollte einen haben.
- Es kann nur einen in jeder Tabelle geben.

- Er kann auch aus mehreren Spalten bestehen, wenn eine Spalte alleine keine Eindeutigkeit schafft.
- Dann wird dieser als **Zusammengesetzter Schlüssel** bezeichnet.

Foreign Key

- Verweist auf den P-Key einer anderen Tabelle.
- Muss in der Referenztabelle vorhanden sein.
- Null werte sind erlaubt (→ Explizit NULL).
- Kaskadierung ist möglich (Änderung- Löschweitergabe).

Unique Key

- Erzwingt die Eindeutigkeit.
- NULL ist erlaubt.
- Ein Index durch schnelles finden wird erstellt.
- Es kann mehrere geben in einer Tabelle.
- Er kann auch aus mehreren Spalten bestehen (Zusammengesetzt).

CHECK

- Werte Prüfung auf Gültigkeit
- Verweise auf Inhalte desselben Datensatzes
- Kein verweis auf andere Datensätze / Tabellen

NOT NULL

- Erzwingt beim einfügen / ändern die eines Wertes.
  - Es darf kein Leerwert (null) eingefügt werden.

Optimierung

Imed Ghaouari

Optimierung (Normalisierung)

- vermeidet Widersprüchliche Inhalte
- vermeidet Anomalien

- Normalformen:
  1. Normalform - Atom
  2. Normalform - Schlüssel
  3. Normalform - kein Schlüssel

- 1NF → 2NF → 3 NF

Erste Normalform (1NF)

- Definition:
  - Jeder Attributwert ist eine atomare (nicht weiter zerlegbare) Dateneinheit und muss frei von Wiederholungsgruppen sein.

- Erläuterung:
  - Zusammengesetzte Inhalte & Mehrfache Inhalte sind zu beseitigen.

1NF

ID	Name
1	Manfred Werner
2	Hans
3	Hans Mueller

ID	Vorname	Nachname
1	Manfred	Werner
2	NULL	Hans
3	Hans	Mueller

ID	Person	Nummer
1	Werner	0221/2353672, 0154/8349825
2	Hans	02203/9303427, 0234/75839
3	Mueller	02345/3539458

PerID	Nummer
1	0221/2353672
1	0154/8349825
2	02203/9303427
2	0234/75839
3	02345/3539458

Zweite Normalform (2NF)

- Definition
  - Die Tabelle muss sich erstmal in der ersten Normalform (1NF) befinden und jedes Nichtschlüsselattribut muss von jedem Schlüsselkandidaten voll funktional abhängig sein.

- Erläuterung:
  - Wenn man **zusammengesetzte Schlüssel** in einer Tabelle hat, dann muss man auf die Nichtschlüsselattribute achten!
  - Jede Spalte, die **nicht ein Schlüssel ist muss** direkt von einem Schlüssel (zusammengesetzten Schlüssel) identfiziert werden können.

2NF

ID	First	Last	City	Zipcode	IdProject	Description	Time
1	Werner	Müller	Cologne	51103	1001	Promotion	120
2	Hans	Otto	Duesseldorf	40742	1003	Specifiation	200
2	Hans	Otto	Duesseldorf	40742	2001	Deployment	235
3	Mueller	Gerner	Bonn	53117	0001	Project Budget	70

- ID → First, Last, City, Zipcode
- IdProject → Description
- ID & IdProject → Time

2NF

ID	First	Last	City	Zipcode
1	Werner	Müller	Cologne	51103
2	Hans	Otto	Duesseldorf	40742
3	Mueller	Gerner	Bonn	53117

IdProject	Description
1001	Promotion
1003	Specifiation
2001	Deployment
0001	Project Budget

ID	IdProject	Time
1	1001	120
2	1003	200
2	2001	235
3	0001	70

3NF

- Definition
  - Die Tabelle muss sich erstmal in der zweiten Normalform (2NF) befinden und jedes Nicht-Schlüssel-Attribut darf von keinem Schlüsselkandidaten transitiv abhängig sein.

- Erläuterung:
  - Jedes Nicht-Schlüssel-Attribut darf nicht von anderen Nicht-Schlüssel-Attribut abhängig sein.

3NF

IdProject	Description
1001	Promotion
1003	Specifiation
2001	Deployment
0001	Project Budget

ID	IdProject	Time
1	1001	120
2	1003	200
2	2001	235
3	0001	70

ID	First	Last	City	Zipcode
1	Werner	Müller	Cologne	51103
2	Hans	Otto	Duesseldorf	40742
3	Mueller	Gerner	Bonn	53117

3NF

ID	First	Last	City	Zipcode
1	Werner	Müller	Cologne	51103
2	Hans	Otto	Duesseldorf	40742
3	Mueller	Gerner	Bonn	53117

ID	First	Last	Zipcode
1	Werner	Müller	51103
2	Hans	Otto	40742
3	Mueller	Gerner	53117

Zipcode	City
51103	Cologne
40742	Duesseldorf
53117	Bonn

SQL

Imed Ghaouari

Structed Query Language (SQL)

- Ist eine standardisierte Abfragesprache für relationale DBS.

- Sie besteht aus 4 Sprachbereiche:
  - Data Definition Language (DDL)
  - Data Manipulation Language (DML)
  - Data Control Language (DCL)
  - Transact Control Language (TCL)

Data Definition Language

- Mit DDL - Anweisungen Erstellt man Tabellen und Beziehungen

- DDL – Anweisungen:
  - CREATE
  - ALTER
  - DROP

- Tabellen bestehen aus verschieden Datentypen.
  - Numerischen Datentypen (Ganzzahlig, Gleit/Fix Komma, Währung)
  - Zeichenfolgen – Datentypen (fixe und variable)
  - Datumstypen

Data Manipulation Language

- Mit DML- Anweisungen erhält man Schreibzugriff auf die DBS.
- Mit DML- Anweisungen erhält man Lesezugriff auf die DBS.

- Schreibzugriff
  - INSERT
  - UPDATE
  - DELETE

- Lesezugriff
  - SELECT

Data Control Language

- Mit DCL - Anweisungen werden Zugriffsberechtigungen geregelt durch die Erstellung und Verwaltung von Benutzern und Rollen.

- DCL-Anweisungen:
  - Grant
  - Deny
  - Revoke

Syntax - 1

Tabelle 6.1: Übersicht über die SQL-Syntax

```
CREATE TABLE <Tabellenname> (<Spaltenname> <Spaltentyp> [<Attributeinschränkung>]
                            {, <Spaltenname> <Spaltentyp> [<Attributeinschränkung>] }
                            [<Tabelleneinschränkung> {, <Tabelleneinschränkung>} ] )

DROP TABLE <Tabellenname>

ALTER TABLE <Tabellenname> ADD <Spaltenname> <Spaltentyp>

SELECT [DISTINCT] <Attributliste>
FROM (<Tabellenname> { <Alias> } | <zusammengesetzte Tabelle>) {, (<Tabellenname>
      { <Alias> } | <zusammengesetzte Tabelle>) }
[WHERE <Bedingung>]
[GROUP BY <Gruppierungsattribute> [HAVING <Gruppenauswahlbedingung> ] ]
[ORDER BY <Spaltenname> [<Ordnung>] {, <Spaltenname> [<Ordnung>] } ]

<Attributliste>::= (* | ( <Spaltenname> | <Funktion> (([DISTINCT] <Spaltenname> | * )))
                        {, (<Spaltenname> | <Funktion> (([DISTINCT] <Spaltenname> | *)) } )
<Gruppierungsattribute>::= <Spaltenname> {, <Spaltenname>}
<Ordnung>::= (ASC | DESC)

INSERT INTO <Tabellenname> [(<Spaltenname> {, <Spaltenname> } ) ]
(WERTE ( <Konstantenwert>, { <Konstantenwert> } ) {, ( <Konstantenwert> {,
<Konstantenwert> } ) } | <SELECT-Anweisung>)
```

Abbildung 6.1 „Quelle: Grundlagen von Datenbanksystemen, Elmasri & Navathe, Pearson Deutschland GmbH, 3. Auflage 2009, S. 185“

Syntax - 2

Tabelle 6.1: Übersicht über die SQL-Syntax

```
DELETE FROM <Tabellenname>
[WHERE <Auswahlbedingung>]

UPDATE <Tabellenname>
SET <Spaltenname> = <Wertausdruck> {, <Spaltenname> = <Wertausdruck> }
[WHERE <Auswahlbedingung>]

CREATE [UNIQUE] INDEX <Indexname>*
ON <Tabellenname> ( <Spaltenname> [ <Ordnung> ] {, <Spaltenname> [ <Ordnung> ] } )
[CLUSTER]

DROP INDEX <Indexname>*

CREATE VIEW <View-Name> [ ( <Spaltenname> {, <Spaltenname> } ) ]
AS <SELECT-Anweisung>

DROP VIEW <View-Name>
```

* Diese beiden letzten Befehle sind nicht Teil des SQL2-Standards.

Abbildung 6.1 „Quelle: Grundlagen von Datenbanksystemen, Elmasri & Navathe, Pearson Deutschland GmbH, 3. Auflage 2009, S. 186“

DML - Lesezugriff

Imed Ghaouari

Reihenfolge der Abarbeitung von SQL-Anweisungen:

1. FROM (Aufruf aller Tabellen)
2. WHERE (Filterung)
3. GROUP BY (Gruppierung)
4. HAVING (Filterung - Gruppen)
5. SELECT (Spalten / Aggregation)
6. ORDER BY (Sortierung)

- Details Siehe folgenden Link:
  - https://docs.microsoft.com/de-de/sql/t-sql/queries/select-transact-sql?view=sql-server-2017

NULL → Ist mit nicht zu vergleichen auch nicht mit sich selbst:

- IS NULL
- IS NOT NULL

Prioritäten - Logische Verknüpfungen & Verneinung:

- NOT
- AND
- OR

Verbunde / Joins

- In Relationale – DBS sind die Daten auf verschiedene Tabellen verteilt, die miteinander in Beziehung stehen.

- Um die Daten wieder zusammen zu setzten muss ich über diese Tabellen joinen (linken).

- Es gibt mehrere Varianten des Joins.
  - INNER JOIN
  - Left / Right JOIN
  - FULL OUTER JOIN
  - CROSS JOIN

Gruppenfunktionen / Aggregatsfunktion

- Mit Ihnen fasst man Werte zusammen und Sie geben deshalb immer nur eine Zeile zurück.

- „NULL“ - Werte werden Ignoriert.

- In der Where-Klausel kann man nicht auf Gruppen Filtern (abarbeitungs-Reihenfolge).

- Bei Gruppen Filtern man mit der Having - Klausel.

Gruppenfunktionen / Aggregatsfunktion - Problem

- Wenn man versucht eine Multimenge und gleichzeitig eine Aggregatsfunktion in derselben Select - Klausel auszuführen schlägt es fehl, da die DBMS nicht weiß, worauf sich die Aggregation beziehen soll.

Lösung

- Alle spalten müssen entweder der Group By - Klausel angehören oder mit einer Aggregatsfunktion versehen sein (Gruppierung oder SUM()).
- Das bedeutet mehrere Spalten in der SELECT-Klausel ohne Gruppenfunktion, müssen in der GROUP BY-Klausel eingeschlossen werden.

Unterabfragen

- Werden dann verwendet, wenn man ansonsten mehrere abfragen hintereinander Ausführern müsste, um das Ergebnis der einen für die Nächste Abfrage zu verwenden.

- Die Unterabfrage wird vor der Hauptabfrage Ausgeführt.

- Sie können auch in der WHERE-Klausel und in der FROM- Klausel eingesetzt werden.

→ Unterabfragen werden in Runden Klammern gesetzt.

SET-Operatoren

- Verbinden mehrere Einzelabfragen vertikal zu einer Gesamtabfrage.

- UNION (ALL)
  - Mit Union bildet man eine Vereinigungsmenge.

- INTERSECT
  - Bildet man die Schnittmenge von beiden zurückgelieferten SELECT Anweisungen.

- EXCEPT
  - Liefert die Ergebnis aus der ersten Abfrage, die in der zweiten nicht vorkommen (Differenzmenge).

SET-Operatoren (Regeln)

- Das Erste „SELECT“ gibt folgendes:
  - Spaltenanzahl & Spaltennamen
  - Reihenfolge
  - Datentypen

- Spaltenanzahl & Datentypen müssen bei allen Anweisungen übereinstimmen.

- WHERE-Klausel wird bei jeder Anweisung Separat definiert.

- ORDER BY wird einmal für alle gemeinsam definiert.

Schema

- Das Schema kann man sich als Strukturierungseinheit, in der sich Tabellen befinden vorstellen.
- Keine Ordnerstruktur → flache (einstufige) Hierarchie
- Tabellen sind Schemaobjekte.
- Ursprünglich war das Schema ein Benutzerbereich der Datenbank indem man Datenobjekte Benutzerbezogen speichern konnte.

→ Organisatorische Trennung & Verwaltung von Berechtigungen

Transaktionen

Imed Ghaouari

Transaktion

- Sind mehrere Anweisungen die als eine Einheit betrachtet werden.

- Entweder werden alle Befehle ausgeführt oder keine.
  (alles oder nichts)

- Beispiel einer Überweisung von Konto A nach Konto B.
  - Für eine Überweisung sind 2 Schritte nötig:
    1. Abbuchung von Konto A.
    2. Einzahlung auf Konto B.

Verwaltung - DBMS

- Transaktionen werden Grundsätzlich vom DBMS gesteuert.
- Wir müssen dem System mitteilen, wann eine Transaktion startet wann sie aufhört und welche Anweisungen dazu gehören.

Implizite - Steuerung

- Die Transaktion startet automatisch bei der ersten DML-Anweisung.
- MySQL / Oracle

Explizite - Steuerung

- Die Transaktion muss für mehrere Anweisungen Explizit gestartet werden.
- MSSQL

Befehle – Transaktionen

COMMIT

- Änderungen werden festgeschrieben.

ROLLBACK

- Änderungen werden verworfen.

SAVEPOINT

- Speichern einen zwischenstand.

Ausführung

Abbildung 12.4: Zustandsübergangsdiagramm mit den Zuständen einer Transaktionsausführung.

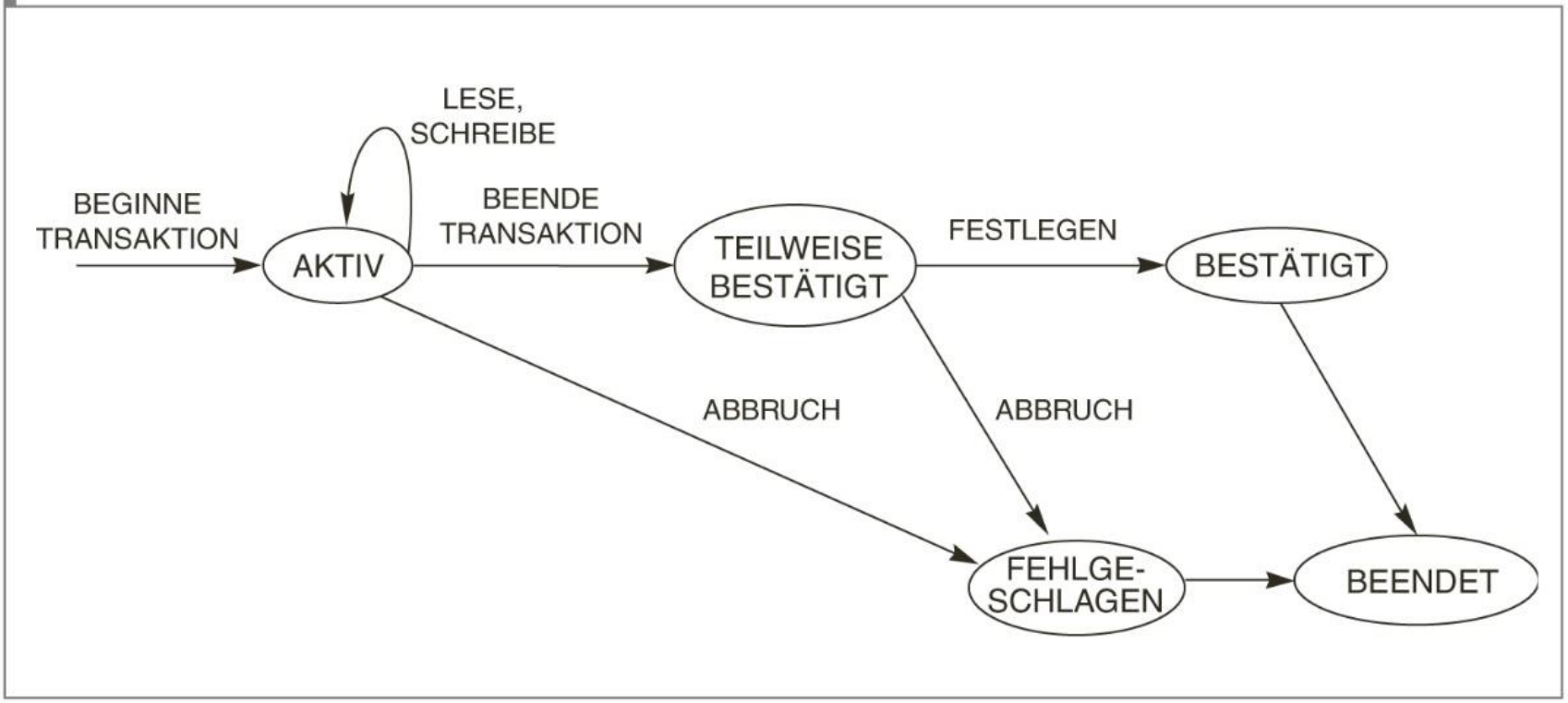


Abbildung 12.4 „Quelle: Grundlagen von Datenbanksystemen, Elmasri & Navathe, Pearson Deutschland GmbH, 3. Auflage 2009, S. 420"

ACID – Prinzip - 1

- Eine Transaktion muss atomar, konsistent, isoliert und dauerhaft sein.

Atomicy (Atomar)

- Es spielt keine Rolle wie viele Schritte die Einheit hat.
- Es gilt „Alles oder nichts“.

Consistency (Konsistent)

- Die Datenbank muss vor und nach einer Transaktion in einem gültigen Zustand verlassen werden.

ACID – Prinzip – 2

Isolation (Isoliert)

- Die Tabellen die an der Transaktion beteiligt sind müssen bis zur beendig gesperrt sein und sorgen, dafür, dass sich andere Transaktionen nicht in die quere kommen.

Durability (Dauerhaft)

- Wenn die Transaktion erfolgreich abgeschlossen wird, dann müssen die Daten Dauerhaft festgeschrieben sein.

SQL - Automatisierung

Imed Ghaouari

Index

- Die Indizierung einer Tabelle, beschleunigt die Suche.
- Es wird dabei eine Baumstruktur mit einem Verweis auf die Daten verwendet.
- Das System verwendet den Index nur bei einer geringen Menge der Indizierten Daten (Wahrscheinlichkeitsschätzung).

- Unterschiedliche Index-Typen variieren je nach Hersteller der DBMS:
  - Gruppierte Index
  - Bitmap Index
  - Hash-Map

Sichten

- auf dem Server gespeicherte Abfragen.
- sie werden von dem Client mit dem Namen aufgerufen.

- Merkmale:
  - bestehen aus einer SELECT – Anweisung.
  - alle Namen müssen eindeutig sein.
  - verwenden Early-Binding (keine Temporären Tabellen).
  - sichten können verschlüsselt werden.
  - sollten keine Sortierung haben.

Gespeicherte Prozeduren (Stored Procedere)

- Auf dem Server gespeicherte Prozeduren.
- Sie werden von dem Client mit dem Namen aufgerufen.

- können die folgenden Merkmale aufweisen:
  - Parameter
  - IF-Anweisungen
  - Schleifen
  - Ergebnismengen (zurück geben)
  - Objekte (erstellen / verändern)
  - Fehelerbehandlung

Funktionen – Skalarwertfunktionen

- Geben nur einen Wert zurück.

- Merkmale:
  - Keine Änderung an Objekte oder Daten.
  - Fehler werden an den User gegeben.
  - Können verschlüsselt werden.
  - Können den User Kontext wechseln.

Funktionen - Tabellenwertfunktionen

- Man kann Sie als Tabellen und Views sehen, die auch Parameter annehmen können.
- Geben eine Tabelle zurück.

- Merkmale:
  - Keine Änderung an Objekte oder Daten.
  - Fehler werden an den User gegeben.
  - Können verschlüsselt werden.
  - Können den User Kontext wechseln.

Mengen

Imed Ghaouari

Definition und Darstellung einer Menge

- Unter einer Menge verstehen wir die Zusammenfassung gewisser, wohlunterschiedener Objekte, Elemente genannt, zu einer Einheit.

- Als Beispiel betrachten wir die Menge der natürlichen Zahlen:

- $\mathbb{N}=\{0,1,2,3,\ldots\}$

Gleiche Mengen

- Zwei Mengen A und B heißen gleich, wenn jedes Element von A auch Element von B ist und umgekehrt.
- A=B (gelesen: A gleich B)

- Als Beispiel betrachten wir zwei Mengen:
  - A= {0,1,2,5,10}
  - B={10,5,2,0,1}

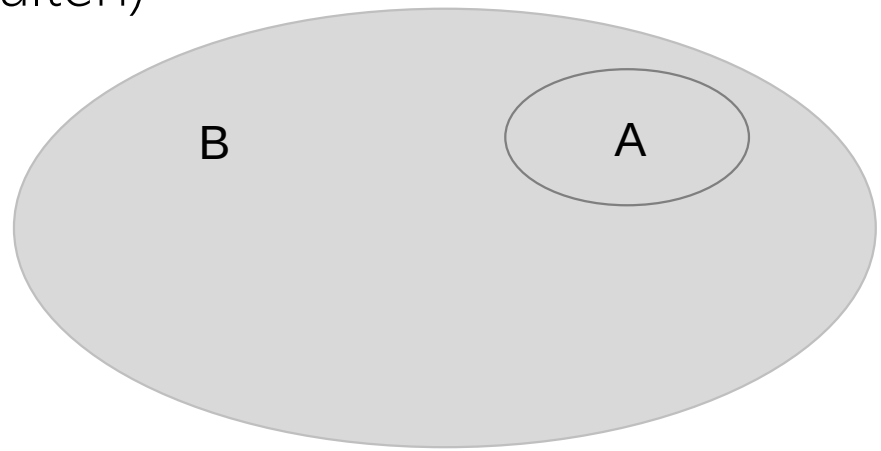
- Jedes Element von A ist auch Element von B und umgekehrt. Die beiden Mengen unterscheiden sich also lediglich in der Anordnung ihrer Elemente und sind daher gleich.

Teilmenge

- Definition:

- Eine Menge A heißt Teilmenge einer Menge B, wenn jedes Element von A auch zur Menge B gehört. Symbolische Schreibweise:

- $A \subset B$ (gelesen: A ist in B enthalten)



Mengenoperationen

- Die Menge algebraischen Operationen sind:

- Schnittmenge

- Differenzmenge

- Vereinigungsmenge

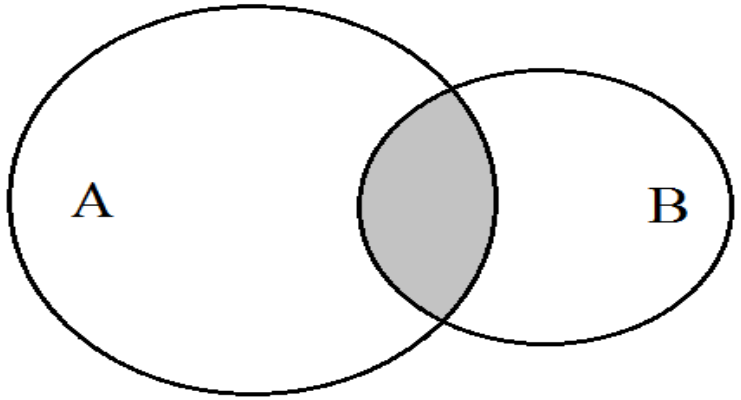
Schnittmenge

- Definition: Die Schnittmenge $A \cap B$ zweier Mengen A und B ist die Menge aller Elemente, die Sowohl zu A als auch zu B gehören:

- Die Schnittmenge $A \cap B$ wird auch als Durchschnitt der Mengen A und B bezeichnet.

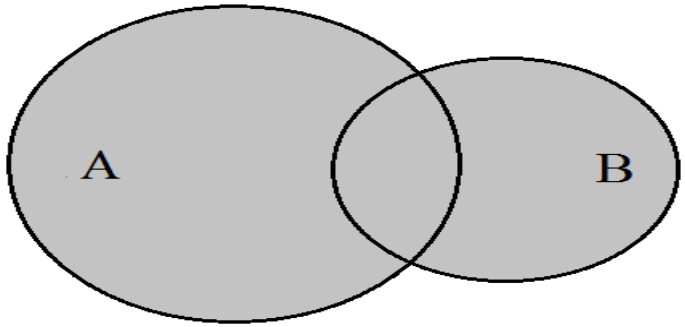
- $A \cap B = \{x | x \in A \text{ und } x \in b\}$

(gelesen: A geschnitten mit B)



Vereinigungsmenge

- Definition: Die Vereinigungsmenge $A \cup B$ zweier Mengen A und B ist die Menge aller Elemente, die zu A oder zu B oder zu beiden Mengen gehören:



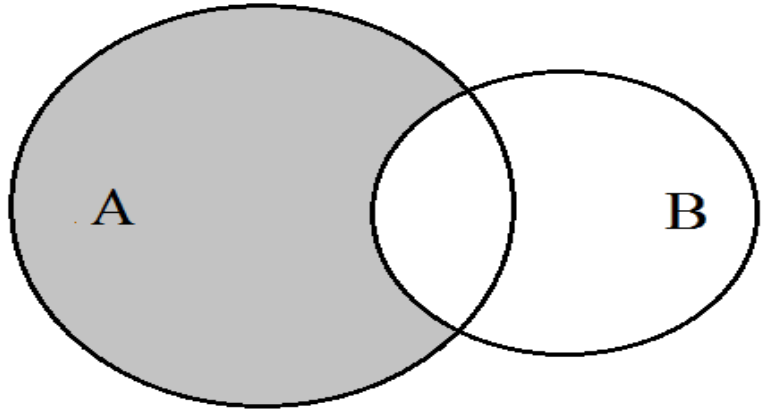
- $A \cup B = \{x | x \in A \quad \text{oder} \quad x \in B\}$
  (gelesen: A vereinigt mit B)

- Man beachte, dass auch diejenigen Elemente zur Vereinigungsmenge gehören, die zugleich Elemente von A und B sind (es handelt sich hier also nicht um das „oder“ im Sinne von „entweder oder“).

Differenzmenge

- Definition: Die Differenzmenge (Restmenge) A\B zweier Mengen A und B ist die Menge aller Elemente, die zu A, nicht aber zu B gehören:

- $A \setminus B = \{x \mid x \in A \quad \text{und} \quad x \notin B\}$ (gelesen: A ohne B)



Literaturangabe

Imed Ghaouari

IT-Handbuch

- IT-Systemelektroniker /-in
- Fachinformatiker /-in
  - Hübscher, Petersen, Rathgeber, Richter, Scharf

Grundlagen von Datenbanksystemen

- Bachelorausgabe
  - Elmasri, Navathe

Datenbanksysteme

- Eine Einführung
  - Kemper, Eickler

Mathematik für Ingenieure und Naturwissenschaftler

- Band 1
  - Lothar Papula

Daten / Informationen / Wissen:

- https://www.artegic.com/de/blog/wo-liegt-der-unterschied-zwischen-daten-informationen-und-wissen/

Datenbanken TU München:

- https://db.in.tum.de/teaching/bookDBMSeinf/

Adventure-Works-Quiz:

- https://sqlzoo.net/wiki/AdventureWorks

Daten / Informationen / Wissen:

- https://www.artegic.com/de/blog/wo-liegt-der-unterschied-zwischen-daten-informationen-und-wissen/

Datenbanken TU München:

- https://db.in.tum.de/teaching/bookDBMSeinf/

Adventure-Works-Quiz:

- https://sqlzoo.net/wiki/AdventureWorks

Übungen:

- http://www.maihack.de/Vorlesungen/datenbanken/sql/Folien/Übungen-Normalisierung_1.pdf
- http://www.is.informatik.uni-kiel.de/~fiedler/teaching/AA/erd/erd_loesungen.pdf

JOIN-Bild:

- https://2.bp.blogspot.com/-oBPhcEuXFA0/VwpQHERiVPI/AAAAAAAAAFsg/r4yUWXmXeQ0ec4YsAGpUTBeGpvS3mUDg/s1600/LEFT%2Bvs%2BRight%2BOuter%2BJoin%2Bin%2BSQL.png

Übungen:

- http://www.maihack.de/Vorlesungen/datenbanken/sql/Folien/Übungen-Normalisierung_1.pdf
- http://www.is.informatik.uni-kiel.de/~fiedler/teaching/AA/erd/erd_loesungen.pdf

JOIN-Bild:

- https://2.bp.blogspot.com/-oBPhcEuXFA0/VwpQHERiVPI/AAAAAAAAAFsg/r4yUWXmXeQ0ec4YsAGpUTBeGpvS3mUDg/s1600/LEFT%2Bvs%2BRight%2BOuter%2BJoin%2Bin%2BSQL.png

Video-DB:

- https://www.youtube.com/watch?v=j_YZXUEtgUo&list=PL8hQ2DvxQbAb09TbiKqsG6WqOGBMcm8Nb
- http://mrbool.com/course/t-sql-step-by-step-course/382