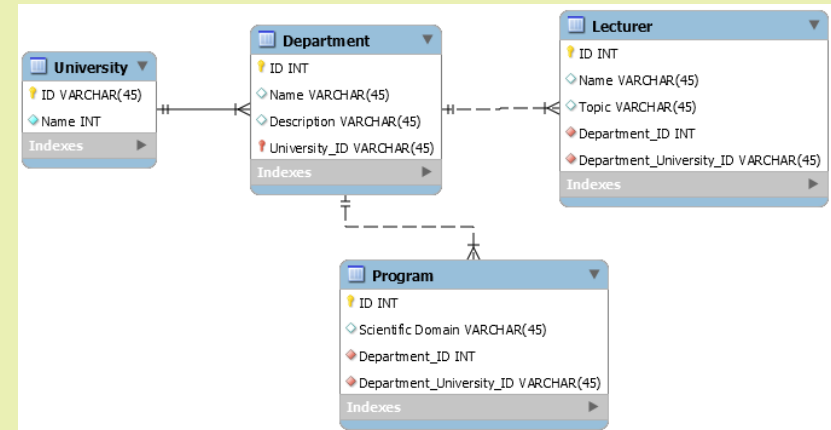


Datenmanagement

Data Requirements – Statische Modellierung

Prof. Dr. Markus Grüne, FB03
Wirtschaftsinformatik



Lernziele

Sie kennen den unterschiedliche Kategorien von Daten (strukturierte Daten, semi-strukturierte und unstrukturierte Daten)

Sie wissen, wie Sie Business Data Objects in einem Data Dictionary beschreiben können.

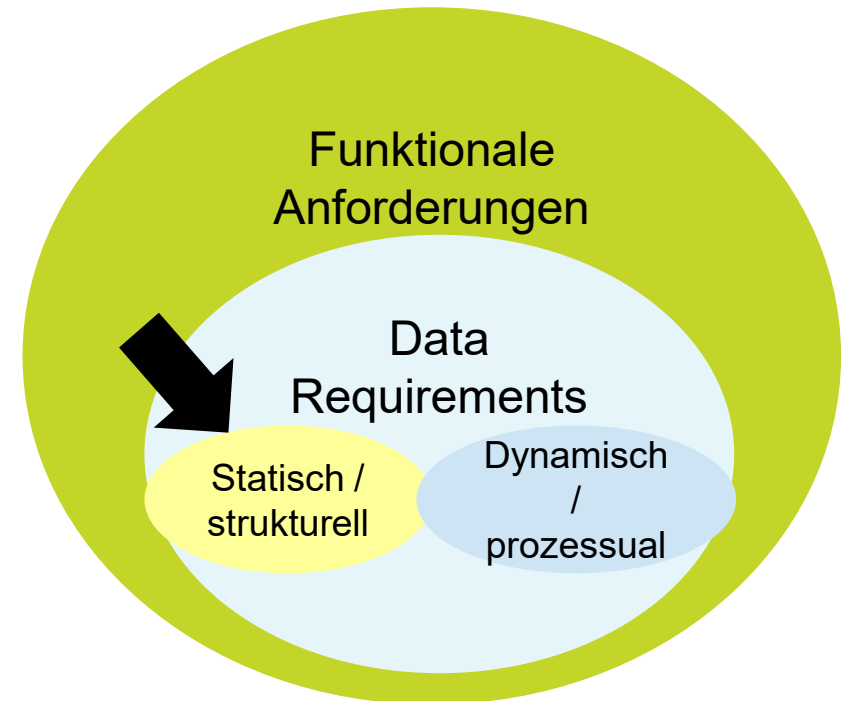
Sie verstehen Entity Relationship-Diagrams und die ER-Modellierung und können diese in ihren Grundzügen anwenden.

Sie wissen, was das relationale Modell ist und welchen Mehrwert dieses bietet.

Data Requirements

Data Requirements sind funktionale Anforderungen, die beschreiben, wie Daten konstruiert und verarbeitet / bewegt werden.

In diesem Kapitel fokussieren wir uns auf die Definition statischer Datenmodelle für die Darstellung von Datenstrukturen.



Datenkategorien – strukturelle Betrachtung

Strukturierte Daten

- Daten haben eine feste Zusammensetzung, z.B. eine feste Reihung, definierte Attribute / Eigenschaften oder vordefinierte Datentypen.

Diese Art von Strukturen finden wir in RDBMS und in Formaten wie CSV.

Der Begriff “Data Mining” wird häufig verwendet, wenn wir die Analyse von strukturierten Daten und die Wissensgewinnung aus strukturierten Daten als Konzept betrachten.

Datenkategorien – strukturelle Betrachtung

Semi-strukturierte Daten

- Haben eine "bestimmte" Struktur, aber passen nicht in relationale oder objektorientierte Datenbankschemata.

Beispiel: Websites, die als XML- oder HTML-Dateien gespeichert werden.

Die Analyse von Websites kann bspw. mittels Verfahren des Web Minings erfolgen.

Datenkategorien – strukturelle Betrachtung

Unstrukturierte Data

Haben entweder kein vordefiniertes Datenmodell oder
Können nicht in relationale Schemata gemappt werden.

Beispiele: Textdokumente, PDF-Dateien, Videos, Bilder, Content in Sozialen Medien.

Andere Arten

Flow Data → z.B. in Video-Streams

Data Dictionary

Während der Anforderungsanalyse zur Gestaltung von Datenbanken, sollte der Fokus zuerst auf der inhaltlichen Erklärung der zu erhebenden Daten liegen. Dabei tritt zu Anfang die technische Umsetzung, d.h. wie die Datenobjekte in der DB gespeichert werden, in den Hintergrund.

Zu Anfang liegt also der Fokus darauf, wie die **Business Stakeholder Datenobjekte verstehen und welche Felder diese Objekte aus Sicht der Fachanwender enthalten sollten.**

Eine Möglichkeit für eine saubere Definition der Anforderungen besteht darin, diese in einem Data Dictionary zu definieren. Achtung: das hier vorgestellte Data Dictionary sollte nicht mit dem technischen Data Dictionary eines DBMS verwechselt werden.

Businessobjekte

- Sind Darstellungen von Realweltobjekten, die die Fachanwender benötigen, um ihre Geschäftstätigkeit zu unterstützen.
- Beispiele: Kreditantrag, Bestellung, Produkt ...

Felder

- Die Eigenschaften oder Attribute, die ein Businessobjekt beschreiben / definieren.
- Beispiel: eine Bestellung hat eine ID, eine Anzahl von bestellten Produkten, eine Versandadresse usw.

Eigenschaften

- Ein Feld hat Eigenschaften (properties), die das Feld genau definieren und Business Rules, die die Belegung des Feldes einschränken.
- Beispiel: das Feld ID ist zusammengesetzt und enthält das Erfassungsdatum der Rechnung

Beispiel Data Dictionary

Property	Description	Example	Notes	Usage
Properties That Define the Business Data Objects and Fields				
ID	A unique identifier for the field. Use a numbering convention that is consistent with the requirements ID numbering	DD001		Necessary
Business Data Object	The name of the business data object that the field is part of.	Customer	The Last Name field is part of the Customer business data object.	Necessary
Field Name	The name the business uses to refer to the field.	Last Name		Necessary
Description	Defines the field. Provides any relevant information beyond the name.	Last Name is a family name or surname of the customer. If the customer only has one name, use the Last Name field.	This is a possible description of the Last Name field.	Optional
Alternate Names	Other names this field is known as. Ideally, you only have one name for each field. However, when you are merging systems or creating a system that is used by multiple groups, a field might have two different, well-established names. If a common name is not clearly understood by all, use this property. This also happens when the names are not synonyms in everyday language but have specific usage within the company. These names can be included in the description if you want to exclude this property.	Family Name	An alternate name for the Last Name field could be Family Name.	Optional
Associated Business Data Object	When a field is another business data object, use this reference and do not repeat the object's information in this row.	Name	Any other business data object. In this case, there might be a Name business data object that has the field's first name, middle name, and last name.	Optional
Data Field	The name under which the data is stored in the system's data store.	LName		Optional
Unique Values?	Whether or not the value for the field has to be unique. This is used if the field is a unique identifier that can be used to differentiate between business data objects of the same type.	No	Multiple customers could have the same last name. This property would be Yes for a field such as social security number.	Optional
Data Type	The type of data used to populate the field. It is best to create and use a set of standard types defined outside an individual Data Dictionary across all of your objects. Also, include formatting information such as patterns for phone numbers or number of decimal digits for real numbers.	Alpha	Basic standard types: Alpha, Numeric, Alphanumeric, or Boolean. More elaborate types: Integer, Real Number, Percent, ZIP/Postal Code, or Phone Number. Alternatively, include formatting information: 3-digit code number, 9-digit number (999.999.9999), or 5 digits plus optional 4 digits (99999- 8888).	Recommended
Length	The maximum number of digits or characters of the field.	50		Recommended

Die Datei basiert auf
[Wiegers and Beatty 2013]
→ Moodle

Das Entity Relationship-Model (ERM)

Das ER-Modell (Entity Relational Model) ist eine Modellierungssprache für die Konstruktion von konzeptuellen Datenmodellen (high level).

Das ERM versucht, Realweltobjekte (oder gedachte Objekte) und die Beziehungen zwischen den Objekten zu beschreiben.

Die ER-Modellierung (der Entwurf von Diagrammen) hilft dabei, Data Requirements zu analysieren und trägt so zum systematischen Entwurf von wohldefinierter Datenbanken bei.

Best Practice: Erst ER-Modell entwerfen, anschließend logisches Modell, dann physisches Modell.

Er-Modell ist die Modellierungssprache

Mit dem ERM werden grafische Repräsentationen der konzeptuellen / logischen Struktur einer Datenbank erstellt.

Mit dem ERM modellieren **Business Analysten** die Entitäten, die in einer Datenbank gespeichert werden und deren Beziehungen auf einer abstrakten Ebene.

Das ER-Diagramm (ERD) ist das Ergebnis der Modellierung.

Es bietet eine Vorausschau über entstehende Tabellen und deren Beziehungen. Die Modellierung erfolgt i.d.R. visuell. Es ist auch möglich, ERDs aus einer Datenbank zurückübersetzen zu lassen (Re-Engineering)

Vom ERD zu Relationen / Tabellen

ERDs können mithilfe von Regeln in relationale Tabellenschemata übersetzt werden. → Beschleunigung des Entwurfsprozesses der DB

Viele DBS unterscheiden zwischen Datenmodellen (ERDs) für logische und physische Modellierung.

Ziele eines ERDs:

- Visualisierung der Anforderungen.
- Kommunikationsmittel zur Kommunikation zwischen unterschiedlichen Stakeholdern, z.B. DB-Admins, Funktionale Analysten, Geschäftsbereiche, QA usw.



Komponenten eines ERDs

Drei grundlegende Konzepte:

- Entities (Entitäten)
- Attribute
- Relationships (Beziehungen)

Beispiel-Entitäten in einer Uni-Datenbank: Student, Course, Lecturer.

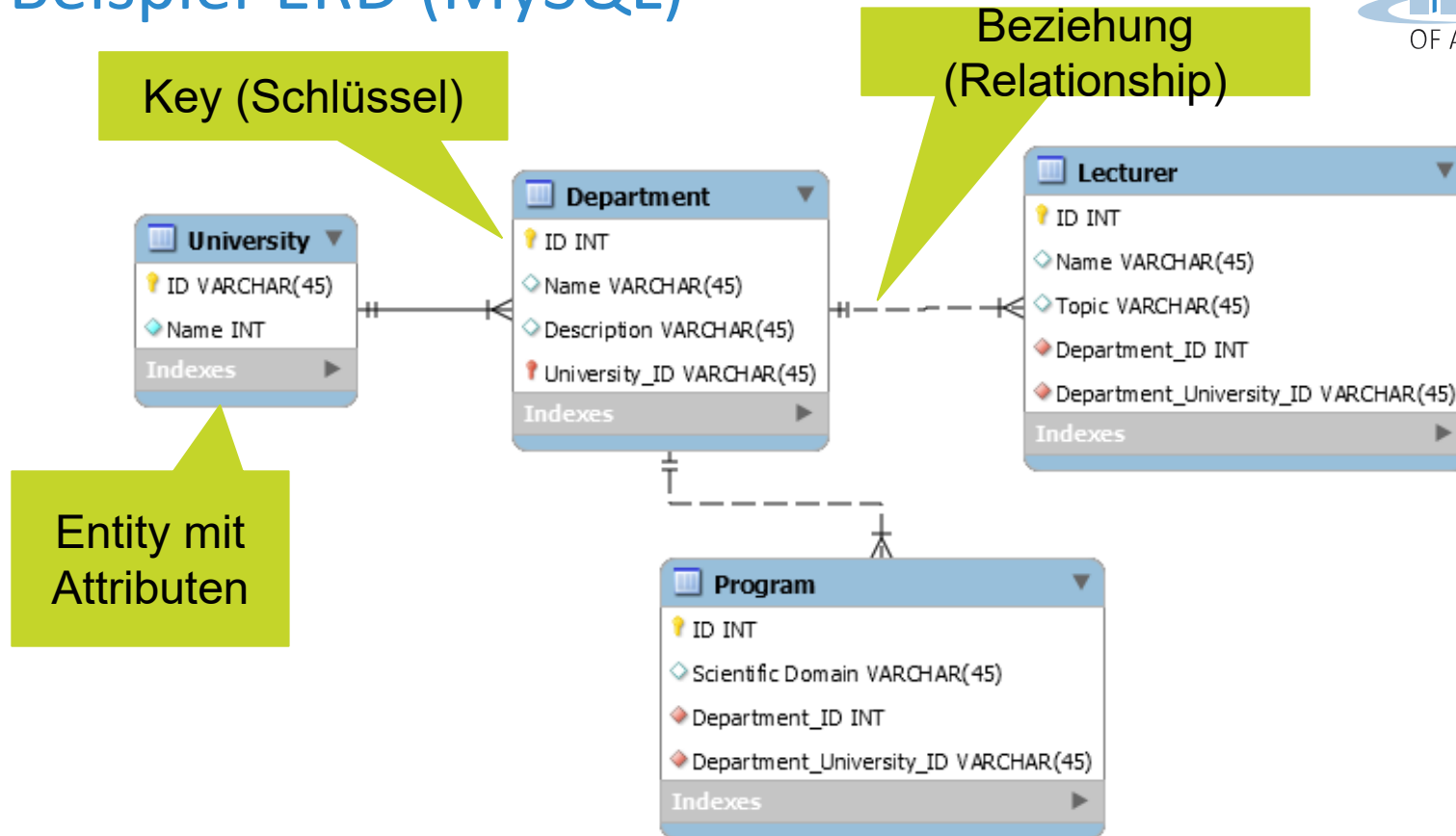
Studierende haben Attribute wie No., Name, DeptID.

Studierende können Beziehungen mit Kursen (Courses) und Dozenten (Lecturers) haben.

University und Department sind 2 Beispiele für Entities (genauer: Entity-Typen).

Jedes Entity hat Attribute University → ID, Name.

Beispiel-ERD (MySQL)



ERD-Beispiel - Erklärungen

Eine University kann mehrere Departments haben.

Alle Departments beschäftigen Lecturers und bieten unterschiedliche Programs an.

Ein Lecturer eines bestimmten Departments lehrt Courses

Jeder Lecturer unterrichtet unterschiedliche Gruppen von Students (fehlt).

Im Kurs verwenden wir die Krähenfußnotation zur Darstellung von ER-Diagramme. Als Alternative findet sich in Lehrbüchern häufig die ursprüngliche Notation nach Chen.

Die Krähenfußnotation (Crow Foot notation) ist die Standardnotation in den gängigen Datenbanksystemen.

Relationships / Beziehungen

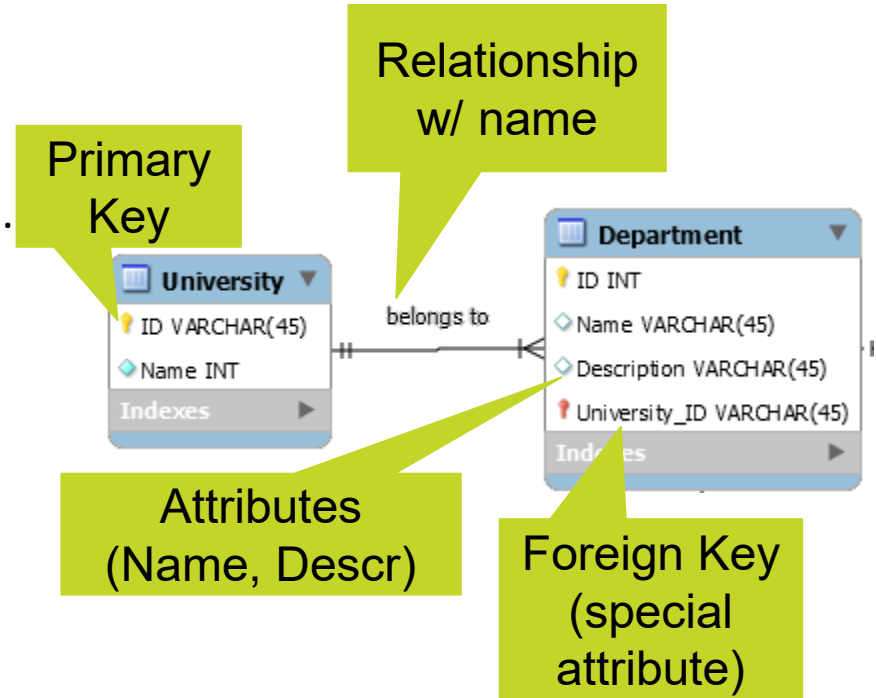
Eine Beziehung ist eine Assoziation zwischen 2 oder mehr Entities.

Die Entities nehmen an einer Beziehung Teil.

Beziehungen können einen Namen tragen.

Eine Beziehung wird durch so genannte Foreign Keys (Fremdschlüssel) definiert (i.d.R. ein Attribut, das auf eine andere Tabelle zeigt).

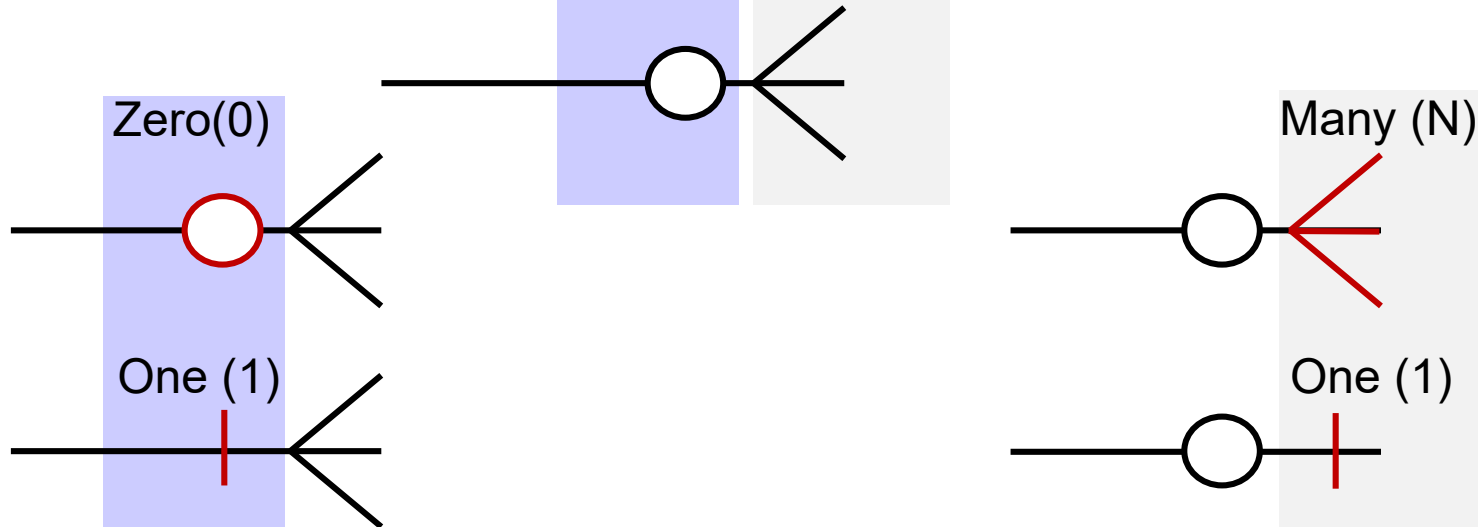
“University_ID” in Department verweist auf die Elemente (Entities) in der Tabelle “University”.



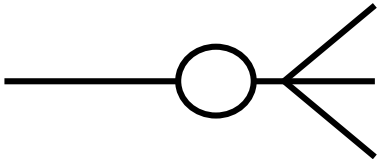
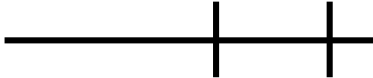

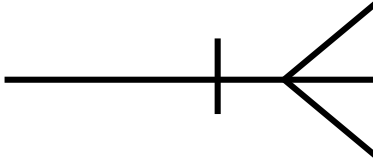
Krähenfußnotation – Cardinality und Modality

Modality minimale Anzahl,
mit der ein Objekt mit
anderen Objekten assoziiert
werden kann

Cardinality maximale Anzahl,
mit der ein Objekt mit anderen
Objekten assoziiert werden
kann.

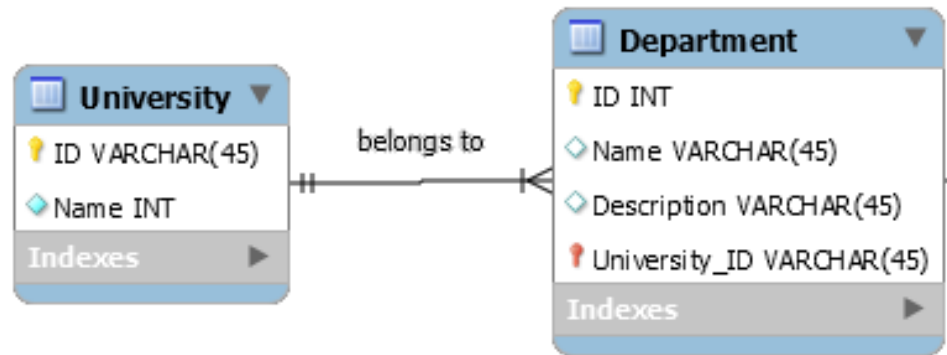


Modalities, Cardinalities in Krähenfußnotation

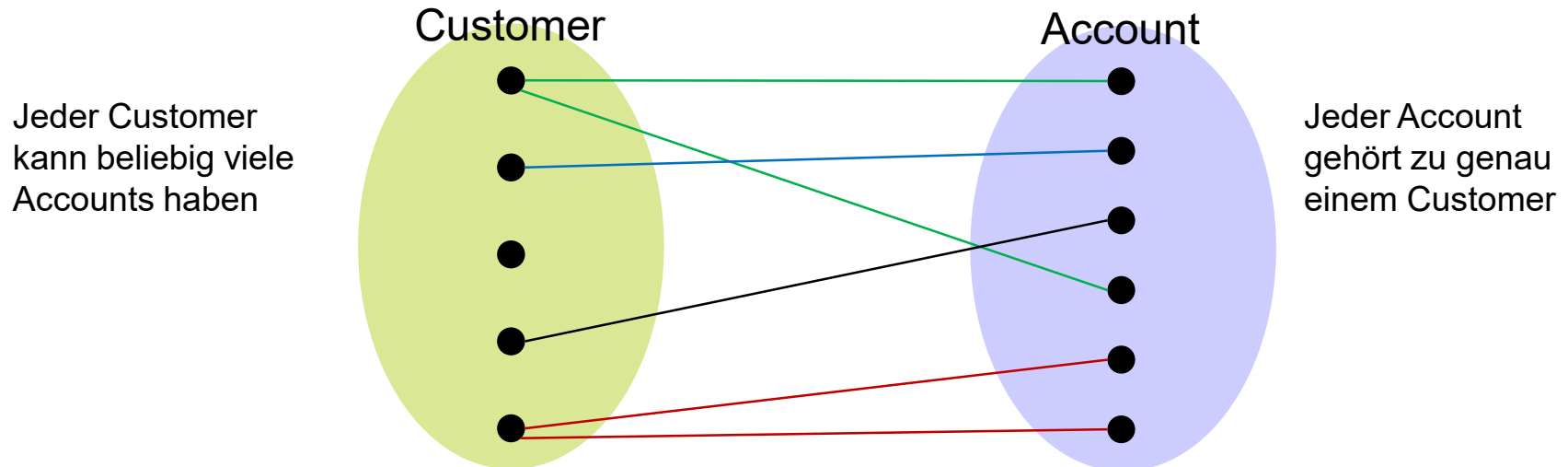
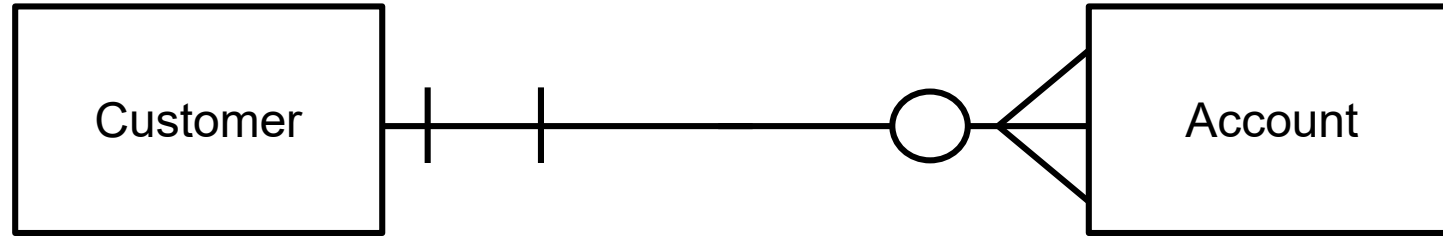
Symbol	Description	Symbol	Description
	Keine oder mehr → N Seite		Eine und nur eine → 1 Seite
	Keine oder eine → 1 Seite		Eine oder mehrere → N Seite

Interpretation von Beziehungen

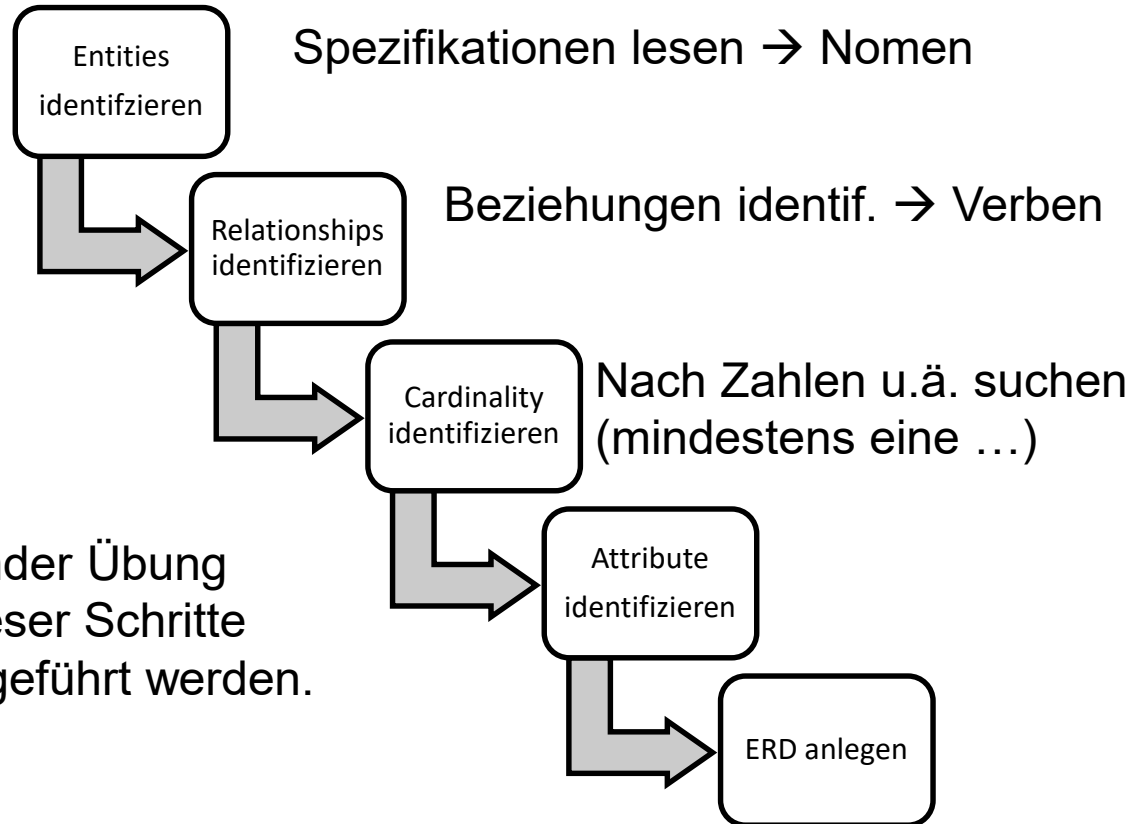
- Von rechts: Ein Department gehört zu minimal einer und höchstens einer University → also genau einer
- Von links: Eine University hat ein oder mehrere Departments.



Interpretation von Beziehungen



Schritte zur Entwicklung eines ERDs



Bei entsprechender Übung können viele dieser Schritte zusammen ausgeführt werden.

Hinweise

Es gibt viele Erweiterungen des ERDs.

Bei Interesse:

- Identifying relationships / Weak Entities
- Is-a-relationships (Aggregation)

Die Konzepte relationaler DB sind wichtig → werden auch für die Modellierung von Data Warehouses später benötigt.

In der Übung werden die Konzepte vertieft.

Das Relationenmodell /relationale Modell

Um die innere Funktionsweise von Datenbanken zu verstehen, müssen wir das relationale Modell verstehen.

Eine Relationale Datenbank enthält eine Sammlung von Relationen. Relationen können als Tabellen gedeutet werden (nicht genau dasselbe!).

- Alle Entitäten eines ERD und (nahezu) alle Relationships können in relationale Strukturen überführt werden.
- Spalten von Tabellen definieren "Attribute" einer Relation
- Zeilen, Tupel, Objekte, Entities definieren die Einträge einer Relation.

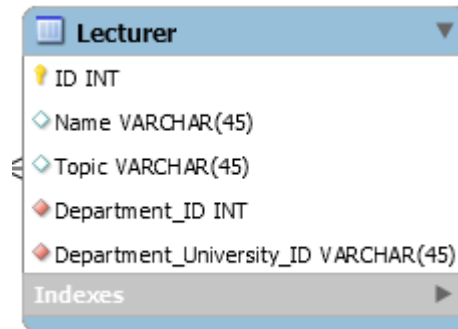
Darstellung einer Relation in einem DBS

Jede Relation hat Attribute und einen Key.

Die Attribute haben einen Datentyp / Domäne.

Frage:

Inwieweit unterscheidet
sich dies von Excel?











Table Name:

Lecturer

Column Name	Datatype	PK	NN	UC
 ID	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
 Name	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 Topic	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 Department_ID	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
 Department_University_ID	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Relationale Modellierung

Das Design eines "guten" Datenmodells ist keine einfache Aufgabe.

Es ist wichtig, nur Informationen in einer Tabelle zu speichern, die zusammen das Entity formen.

In einer Tabelle „Cars“ würden Sie bspw. nicht die Personen speichern, die das repräsentierte Auto gefahren haben, wohl aber die PS-Zahl.

Es gibt einige, etablierte Regeln zur Definition "guter" Tabellenstrukturen. Diese werden unter dem Begriff der "Normalisierung" zusammen gefasst.

Anomalien – Einfügeanomalie

ID	Name	DOB	Place of Residence	DeptNo	DeptName	DeptMgr
1	Theo Rhetic	12-03-1986	Wetzlar	1	Logistics	Schmitt
2	Sophia Hagia	23-06-1976	Istanbul	1	Logistics	Schmitt
3	Paris Dijon	11-11-1995	Moskow	2	Service	Werner
4	Yevgenij Syrtchuk	23-02-1987	Patna	1	Logistics	Schmidt
				3	Development	Wolf
5	He Hu Must Notbenamed	03-09-1977	Death Valley	3	Development	Wolf

Anomalien – Updateanomalie

ID	Name	DOB	Place of Residence	DeptNo	DeptName	DeptMgr
1	Theo Rhetic	12-03-1986	Wetzlar	1	Logistics	Schmitt
2	Sophia Hagia	23-06-1976	Istanbul	1	Logistics	Schmitt Kumar
3	Paris Dijon	11-11-1995	Moskow	2	Service	Werner

Anomalien – Löschanomalie

ID	Name	DOB	Place of Residence	DeptNo	DeptName	DeptMgr
1	Theo Rhetic	12-03-1986	Wetzlar	1	Logistics	Kumar
2	Sophia Hagia	23-06-1976	Istanbul	1	Logistics	Kumar
3	Paris Dijon	11-11-1995	Moskow	2	Service	Werner

Anomalien – Einfügeanomalie

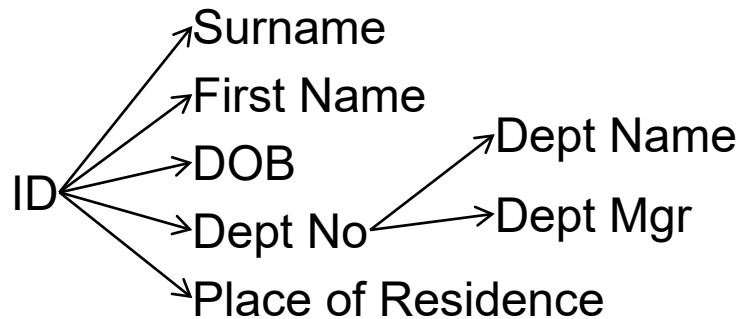
ID	Name	DOB	Place of Residence	Degrees	DeptNo	DeptName	DeptMgr
1	Theo Rhetic	12-03-1986	Wetzlar	B.A., M.Sc.	1	Logistics	Kumar
2	Sophia Hagia	23-06-1976	Istanbul	B.A.	1	Logistics	Kumar
3	Paris Dijon	11-11-1995	Moskow	B.Sc.	2	Service	Werner
4	Yevgenij Syrtchuk	23-02-1987	Patna	B.Sc., PMP, B.A.	1	Logistics	Kumar

Erste Normalform (1NF)



ID	Surname	First Name	DOB	Place of Residence	Dept No	DeptName	DeptMgr																						
1	Rhetic	Theo	12-03-1986	Wetzlar	1	Logistics	Kumar																						
2	Hagia	Sophia	23-06-1976	Istanbul	1	Logistics	Kumar																						
3	Dijon	Paris	11-11-1995	Moskow	<table><tr><th>ID</th><th>PersID</th><th>Degree</th><th>Year</th></tr><tr><td>1</td><td>1</td><td>B.A.</td><td>2006</td></tr><tr><td>2</td><td>1</td><td>M.Sc.</td><td>2008</td></tr><tr><td>3</td><td>2</td><td>B.A.</td><td>2007</td></tr><tr><td>4</td><td>3</td><td>B.Sc.</td><td>2010</td></tr><tr><td>...</td><td>...</td><td>...</td><td>...</td></tr></table>	ID	PersID	Degree	Year	1	1	B.A.	2006	2	1	M.Sc.	2008	3	2	B.A.	2007	4	3	B.Sc.	2010
ID	PersID	Degree	Year																										
1	1	B.A.	2006																										
2	1	M.Sc.	2008																										
3	2	B.A.	2007																										
4	3	B.Sc.	2010																										
...																										
4	Syrtchuk	Yevgenij	23-02-1987	Patna																									

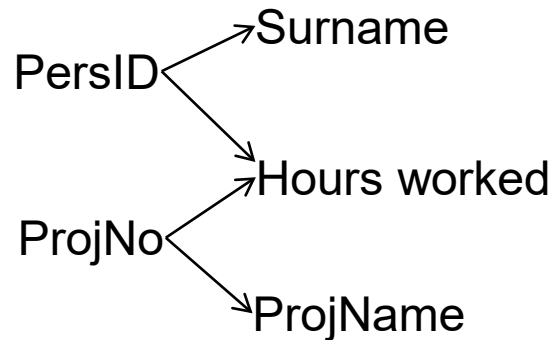
Zweite Normalform (2NF)

ID	Surname	First Name	DOB	Place of Residence	Dept No	DeptName	DeptMgr
1	Rhetic	Theo	12-03-1986	Wetzlar	1	Logistics	Kumar
2	Hagia	Sophia	23-06-1976	Istanbul	1	Logistics	Kumar
3	Dijon	Paris	11-11-1995	Moskow	2	Service	Werner
4	Syrtchuk	Yevgenij	23-02-1987	Patna	1	Logistics	Kumar



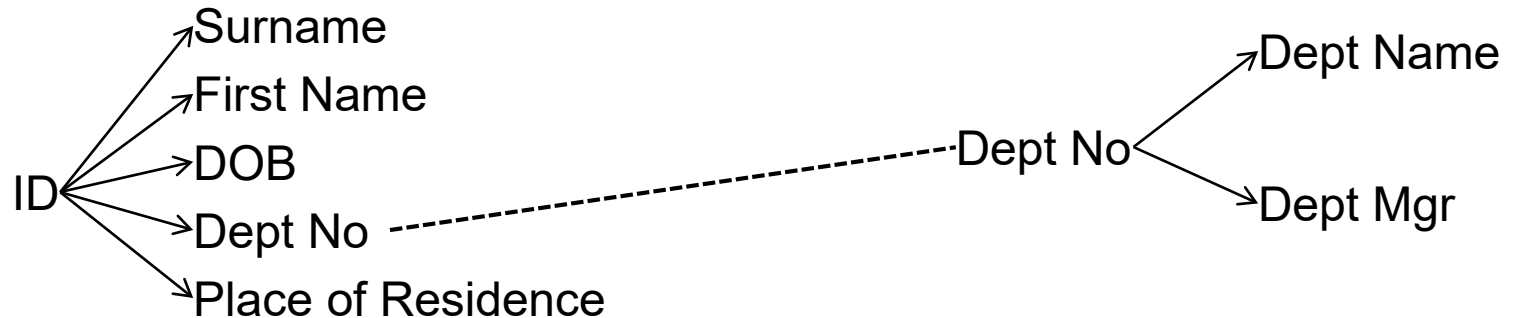
2NF

PersID 	Surname	ProjNo 	ProjName	Hours worked
1	Rhetic	10	Intranet	30
2	Hagia	10	Intranet	45
3	Dijon	21	Extranet	90
3	Dijon	10	Intranet	5

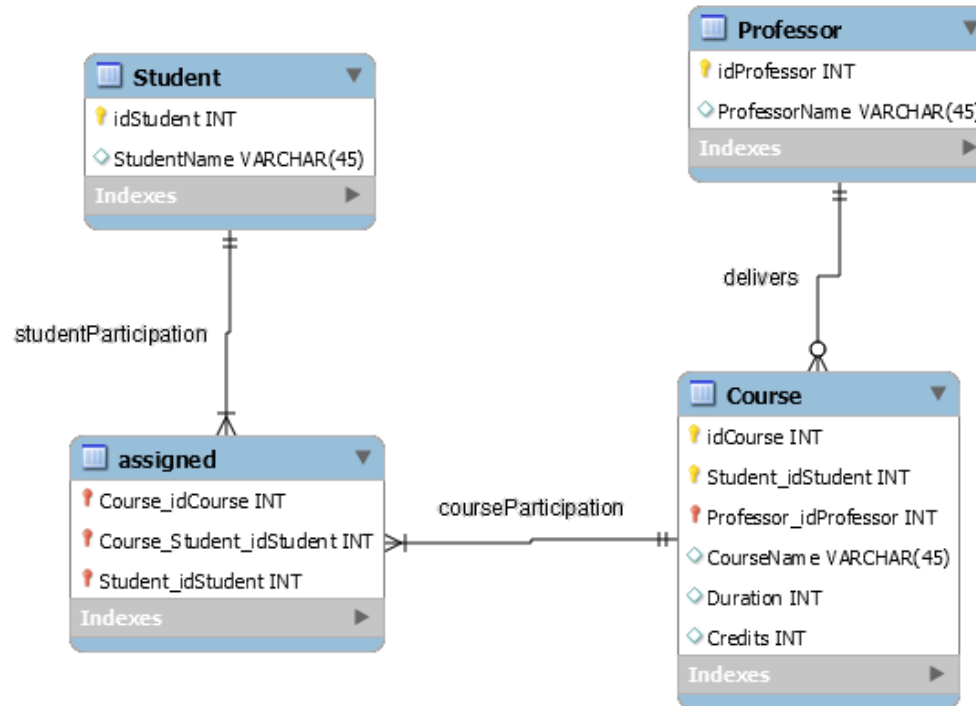


Dritte Normalform (3NF)

ID	Surname	First Name	DOB	Place of Residence	Dept No	Dept No	DeptName	DeptMgr
1	Rhetic	Theo	12-03-1986	Wetzlar	1	1	Logistics	Kumar
2	Hagia	Sophia	23-06-1976	Istanbul	1	2	Service	Werner
3	Dijon	Paris	11-11-1995	Moskow	2			
4	Syrтчuk	Yevgenij	23-02-1987	Patna	1			



Noch ein Blick auf die Datenbank (zum Schluss)



Key Takeaways

In diesem Kapitel haben wir uns vorwiegend strukturierte Daten angesehen.

Wir haben Verfahren kennen gelernt, um strukturierte Daten zu dokumentieren (Data Dictionary) und zu modellieren / visualisieren (ERD).

Wir haben einen Blick auf das relationale Modell geworfen und kennen gelernt, welche Qualitätsmerkmale Relationen erfüllen sollten (Normalformen)

WIEGERS, K., AND BEATTY, J. 2013. *Software Requirements*. Microsoft Press.