Hochschule Karlsruhe
University of
Applied Sciences

Hochschule Karlsruhe – Data Engineering
WS 2021/2022
DSCB330 – Vorlesung 10 – Data Warehouse 1

Foto: Quirin Leppert

Themenübersicht

Data Integration

- Data Formats (csv, XML, json)
- Extract, Transform, Load
- Object Relation Mapper (ORM)
- Staging

Data Processing

- Relationale Datenbanken
- nicht-relationale Datenbanken
- Resource Description Framework (RDF)
- Ontologien
- **Data Warehouse**

Data Modelling

- Serialisierung
- OPC UA
- MQTT
- Pub/Sub
- Data pipelines
  - Apache Airflow
  - gRPC

Web-Service Architektur

- Front-End
- Backend for Frontend (BFF)
- Micro Services
- Docker Container

Security

- Security ist wichtig in allen Phasen der Softwareentwicklung und Datenbereitstellung.

Übung

- Erstellung eines Daten-Modells einer prozesstechnischen Anlage
- Statische Daten
- Dynamische Daten
- Auswertung der Daten

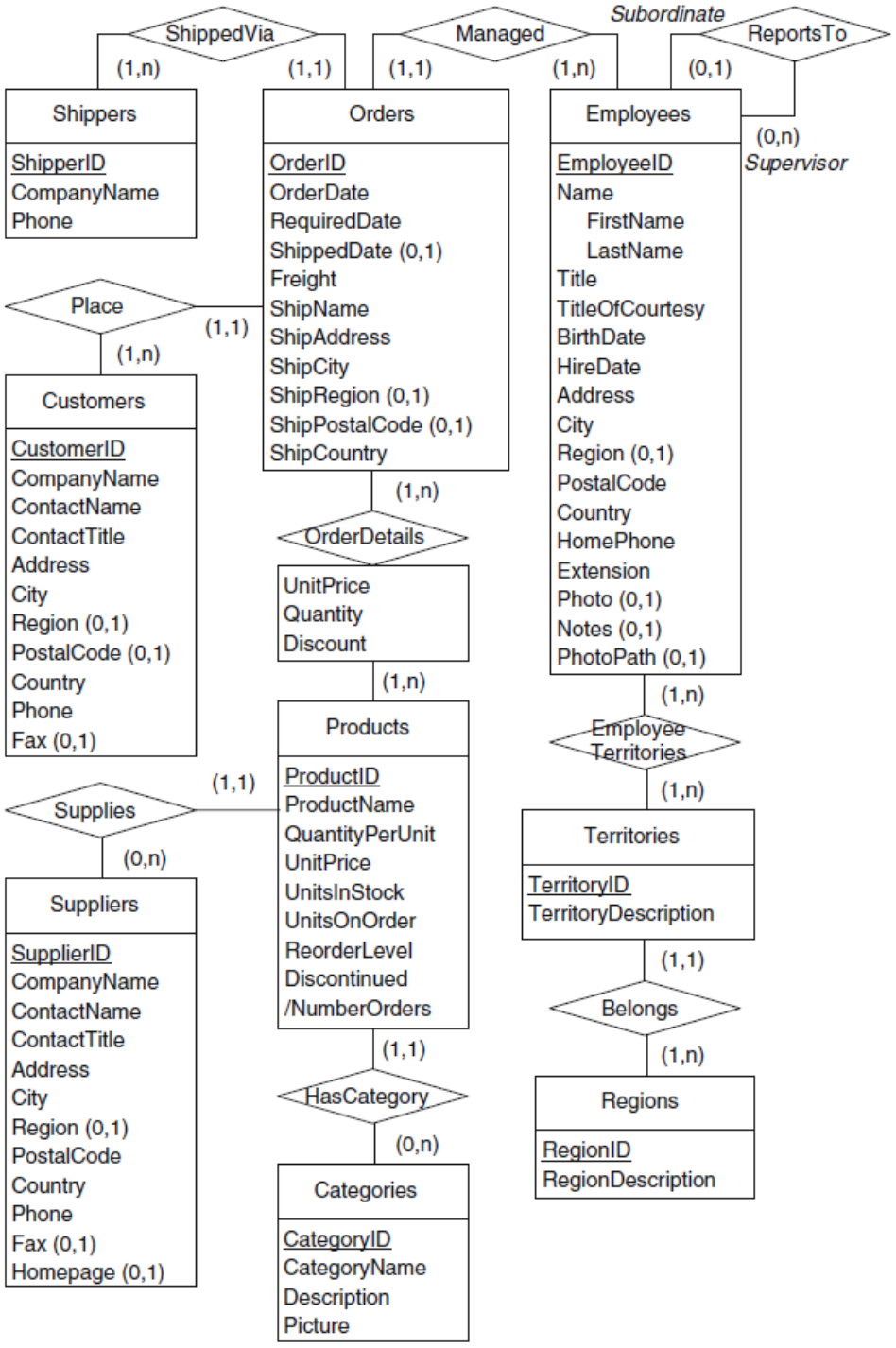
Database Concepts
Conceptual Design

Conceptual Design

- + user-oriented representation of the database
- + does not contain any implementation considerations
- + Modellierung als entity-relationship model

Entity-relationship model

- + **Entity types** repräsentieren real-world objects.
- - Beispiel aus der Northwind-Datenbank: Employees, Orders, Customers
- + **Entity/Instance**: Object eines Entity Types
- + **Population**: set of entities/instances
- + **Relationship types** beschreiben Beziehungen zwischen Objekten, z.B. Supplies, ReportsTo, HasCategory
- + **Role**: Rolle in einer Relationship
- + **Binary**: Beziehung zwischen **zwei** Objekt-Typen (Abhängig von der Kardinalität one-to-one, one-to-many, many-to-many)
- + **n-ary** für Beziehungen zwischen mehreren Objekt-Typen



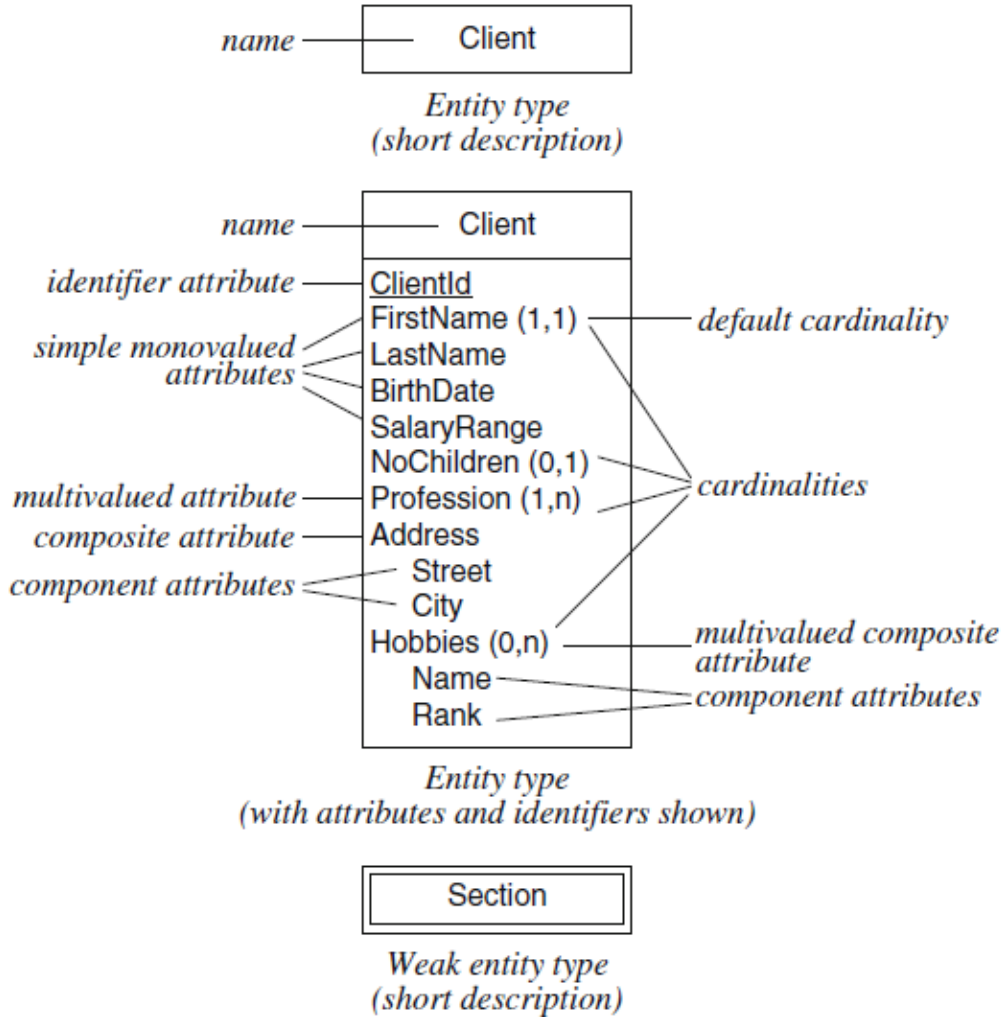
Entity-relationship-Modell der Northwind-Datenbank

Quelle: A. Vaisman and E. Zimányi, Data Warehouse Systems, Data-Centric Systems and Applications, DOI 10.1007/97

Database Concepts
Conceptual Design

Entity-relationship model

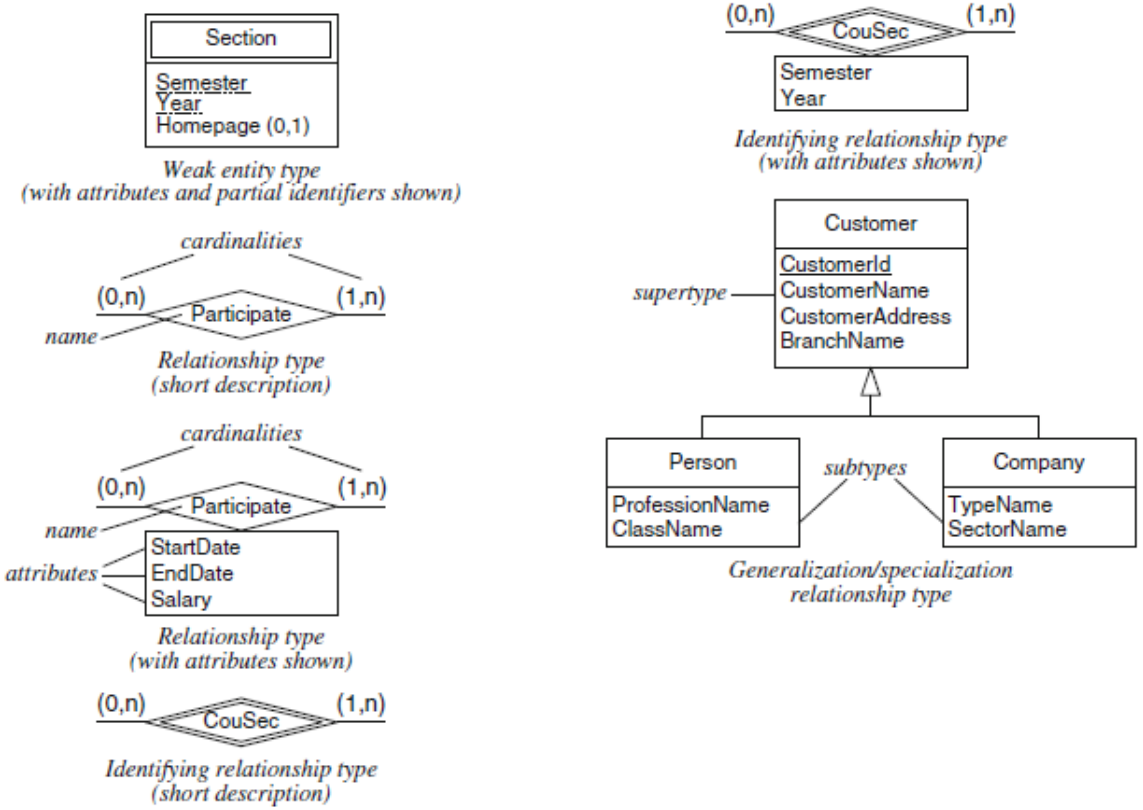
- **Cardinality** (Kardinalität) beschreibt die minimale/maximale Anzahl einer Entität in einer Beziehung.
- **Optional** role: Cardinality = 0
- **Mandatory** role: Cardinality > 0
- **Monovalued** role: Cardinality = 1
- **Multivalued** role: Cardinality > 1
- Ein entity type kann rekursiv (**recursive**) sein, z.B. ReportsTo
- **Weak entity** types haben keinen eigenen Identifier und hängen von anderen entity types (owner) ab.
- **identifying relationship type** ist die Beziehung einer weak entity zum owner.
- Eine normale Beziehung (nicht identifizierend) ist ein **regular relationship type**.



Quelle: A. Vaisman and E. Zimányi, Data Warehouse Systems, Data-Centric Systems and Applications, DOI 10.1007/978-3-642-54655-6, © Springer-Verlag Berlin Heidelberg 2014

Database Concepts
Conceptual Design

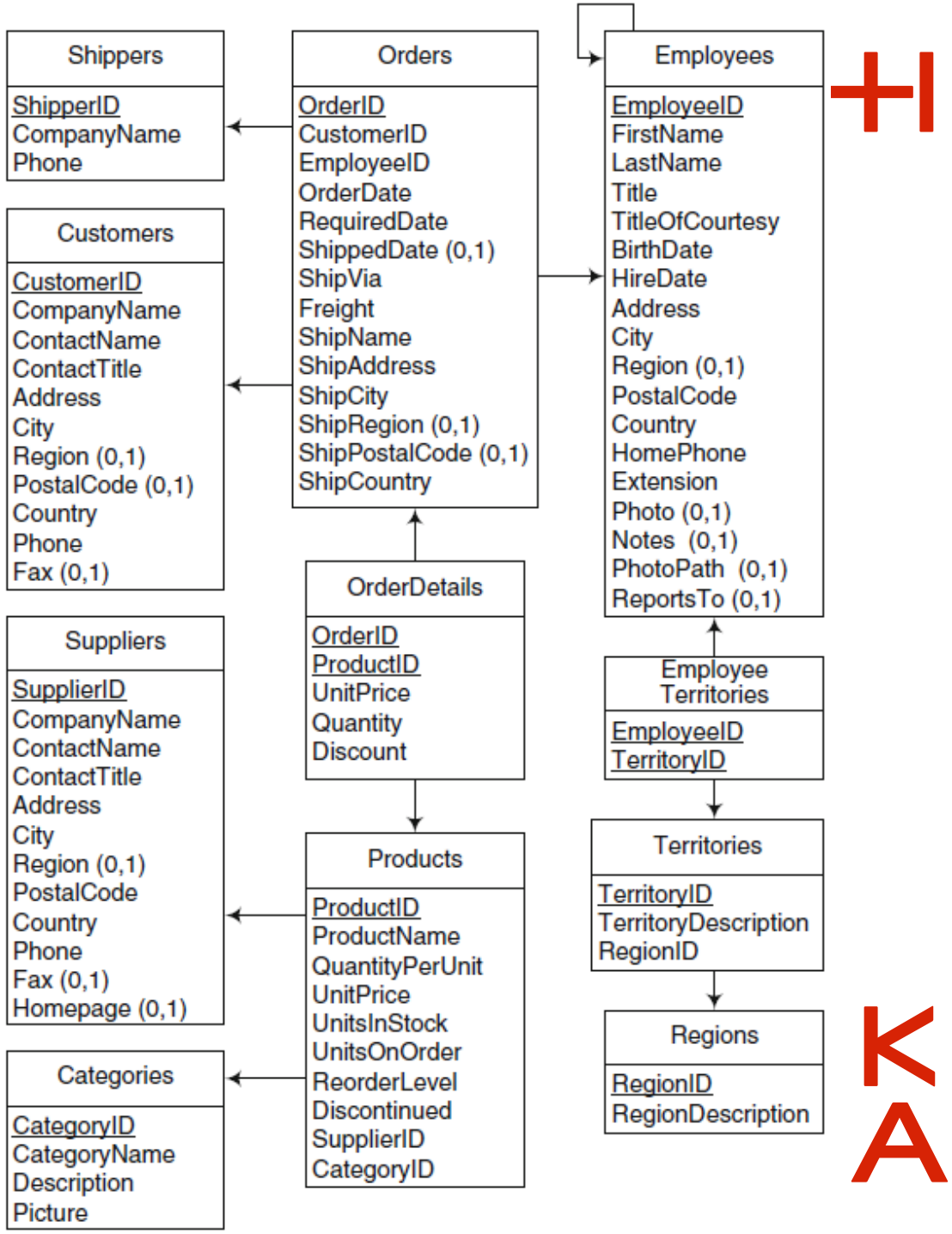
Entity-relationship model



Database Concepts
Logical Database Design

Relational model (relationales Datenmodell)

- Beschreibt die Beziehungen in einer relationalen Datenbank
- Überführung eines ERM in ein relationales Datenmodell
- Jeder Entitätstyp wird im Relationenmodell zu einer eigenständigen Relation.
- Jeder komplexe Beziehungstyp (n:m-Beziehung) wird ebenfalls eine eigenständige Relation, wobei die Tabelle einen aus allen involvierten Entitätstypen zusammengesetzten Primärschlüssel erhält.
- Die Abbildung einfacher Beziehungstypen (1:n-Beziehung) erfordert keine eigenständige Relation. Bei der hier verwendeten Kardinalitätsnotation wird der Primärschlüssel des Entitätstypen mit der „n-Beziehung“ als Attribut in den Entitätstypen mit der „1-Beziehung“ eingebunden.

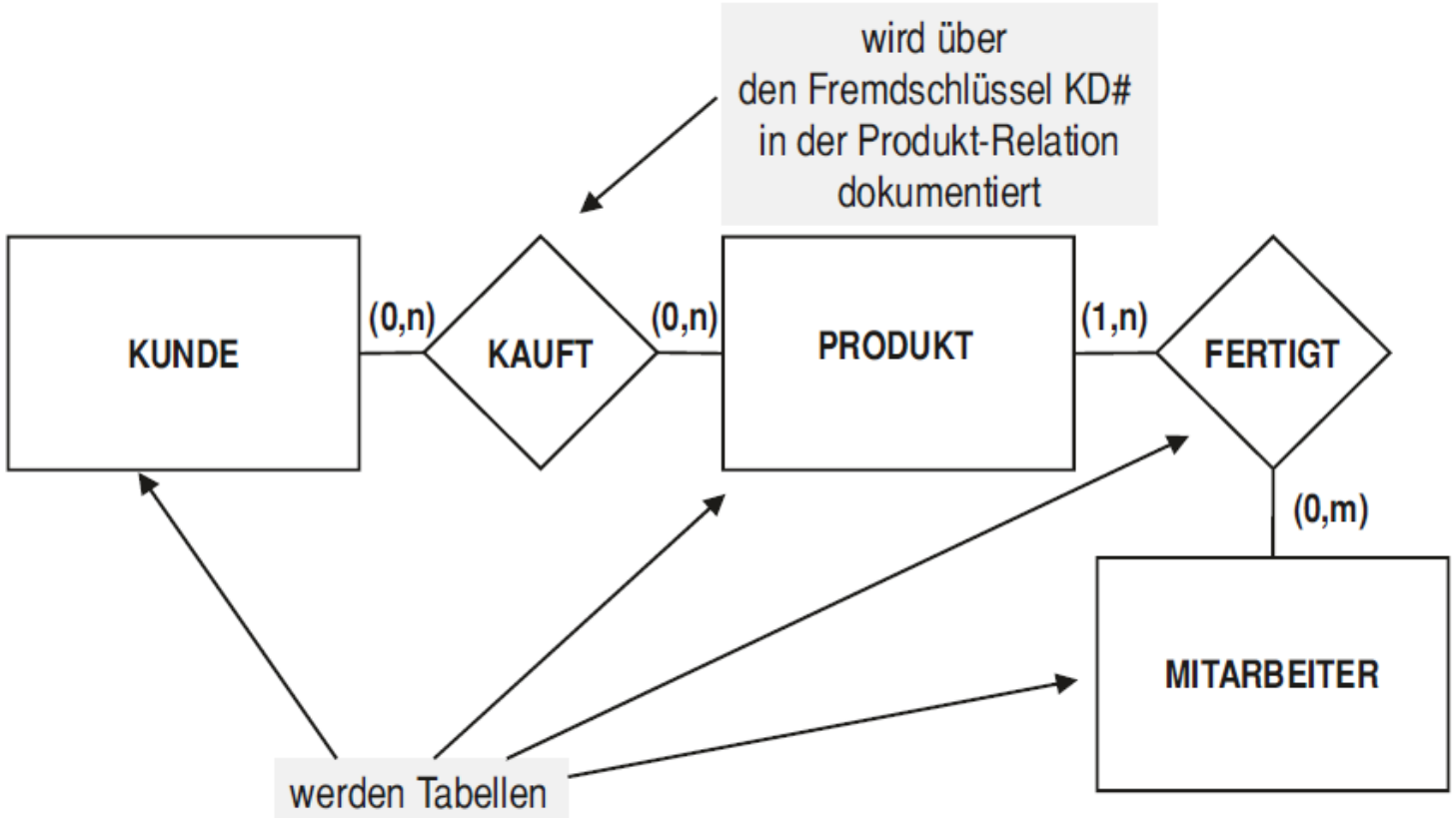


Relational model
der Northwind-Datenbank

Quelle: A. Vaisman and E. Zimányi, Data Warehouse Systems, Data-Centric Systems and Applications, DOI 10.1007/978-3
Baars, Kemper, Business Intelligence & Analytics – Grundlagen und praktische Anwendungen, 4. Auflage, SpringerVieweg

Database Concepts
Logical Database Design

Beispiel ERM → Relational Model



Relationen:

KUNDE(KD#, Name, ...)

PRODUKT(PR#, Preis, ..., KD#)

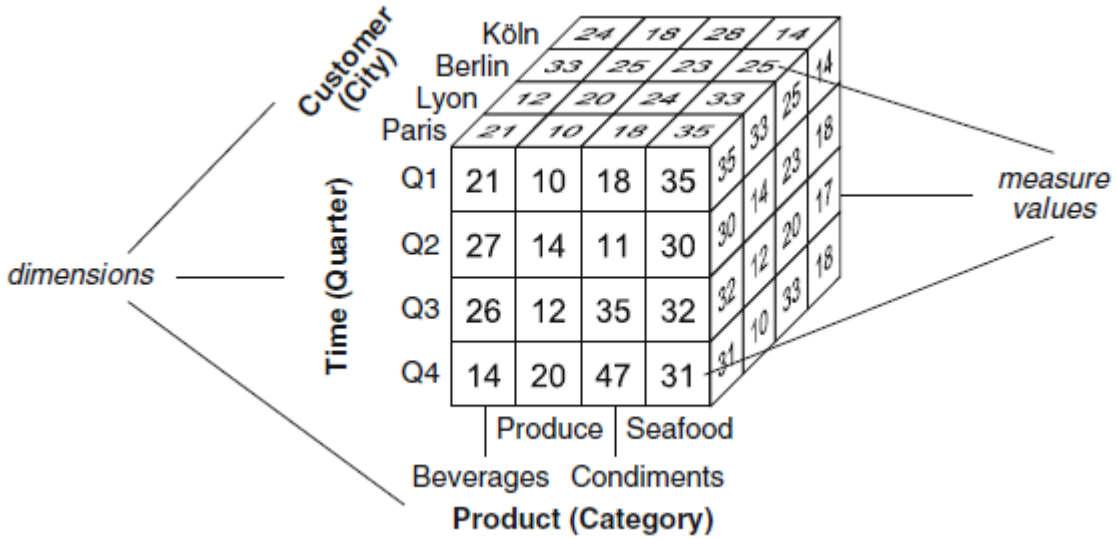
MITARBEITER(MA#, Name, ...)

FERTIGT(PR#, MA#, Arbeitstunden, ...)

Quelle: Baars, Kemper, Business Intelligence & Analytics – Grundlagen und praktische Anwendungen, 4. Auflage, SpringerVieweg, 2021

Data Warehouse Concepts
Multidimensional Model

+ **OLAP**: online analytical processing
+ Beantworten von Fragestellungen wie
- Welcher Umsatz wurde im Juni 2010 in der Region Ost mit dem Produkt C220 bei dem Kundentyp Privatkunden erzielt?
+ Multidimensional Model: Daten werden multi-dimensional dargestellt, als **data cube** oder **hypercube**.
+ Ein data cube besteht aus **dimensions** und **facts**.
+ **Fakten** – mit assoziierten numerischen Werten (Measures) – sind numerische Werte, die den Mittelpunkt der Datenanalyse bilden. Aus semantischer Sicht stellen Fakten fachlich definierte Kennzahlen dar.
+ Dimensionen sind Sichtweisen (perspectives) zur Analyse der Daten.
+ Dimension level repräsentiert die Granularität (level of detail).
+ Beispiel: Cube für Verkaufszahlen basierend auf der Northwind Database.
- Drei Dimensionen: Product, Time, Customer.
- Aggregation: Product nach Kategorie, Time nach Quarter, Customer nach der Stadt
+ Instanzen einer Dimension werden **member** genannt.
- Beispiel: Seafood and Beverages sind member der Dimension Product
- **Facts / cells** eines data cubes haben assoziierte numerische Werte (**measures**).



Quelle: A. Vaisman and E. Zimányi, Data Warehouse Systems, Data-Centric Systems and Applications, DOI 10.1007/978-3-642-54655-6, © Springer-Verlag Berlin Heidelberg 2014
Baars, Kemper, Business Intelligence & Analytics – Grundlagen und praktische Anwendungen, 4. Auflage, SpringerVieweg, 2021

Data Warehouse Concepts
Multidimensional Model

Notation of the MultiDim model

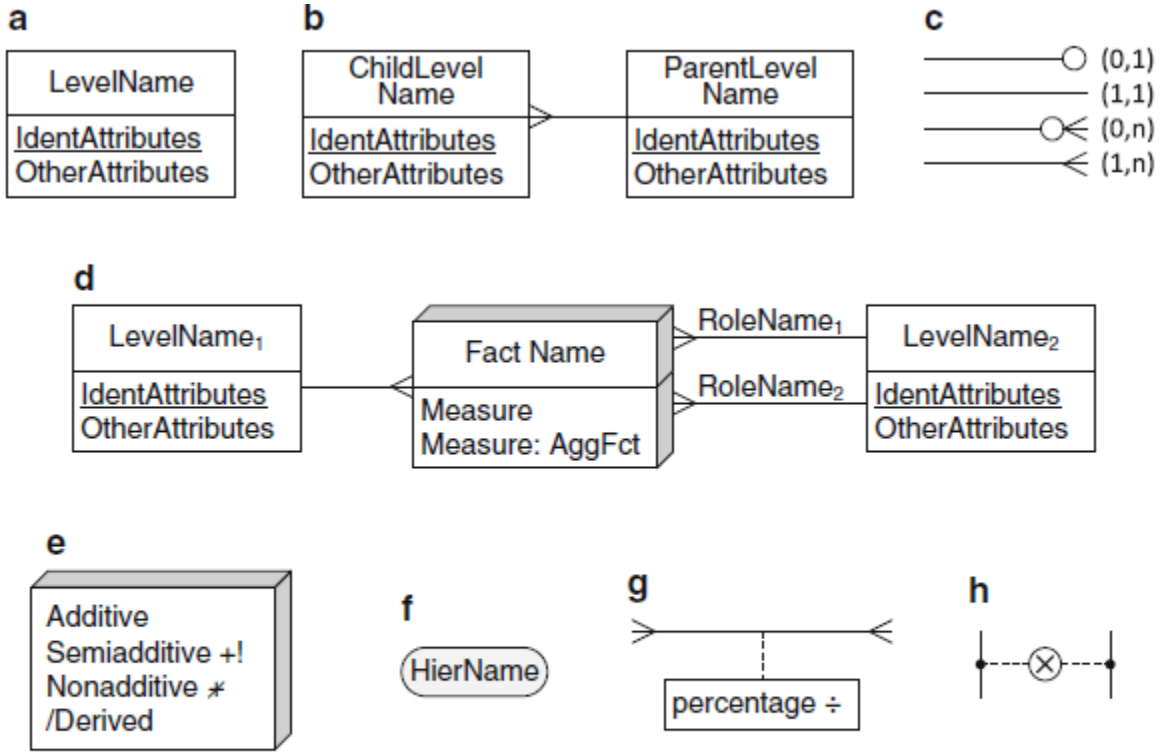
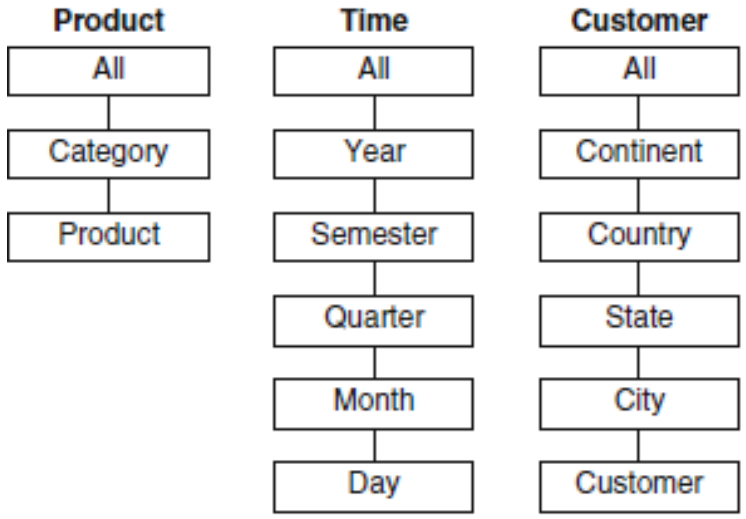


Fig. 4.1 Notation of the MultiDim model. (a) Level. (b) Hierarchy. (c) Cardinalities. (d) Fact with measures and associated levels. (e) Types of measures. (f) Hierarchy name. (g) Distributing attribute. (h) Exclusive relationships

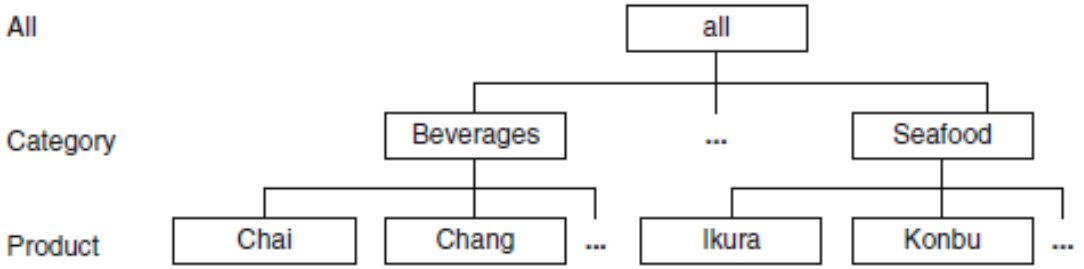
Quelle: A. Vaisman and E. Zimányi, Data Warehouse Systems, Data-Centric Systems and Applications, DOI 10.1007/978-3-642-54655-6, © Springer-Verlag Berlin Heidelberg 2014

Data Warehouse Concepts
Multidimensional Model

- **Hierarchien**



- **Members einer Hierarchie**

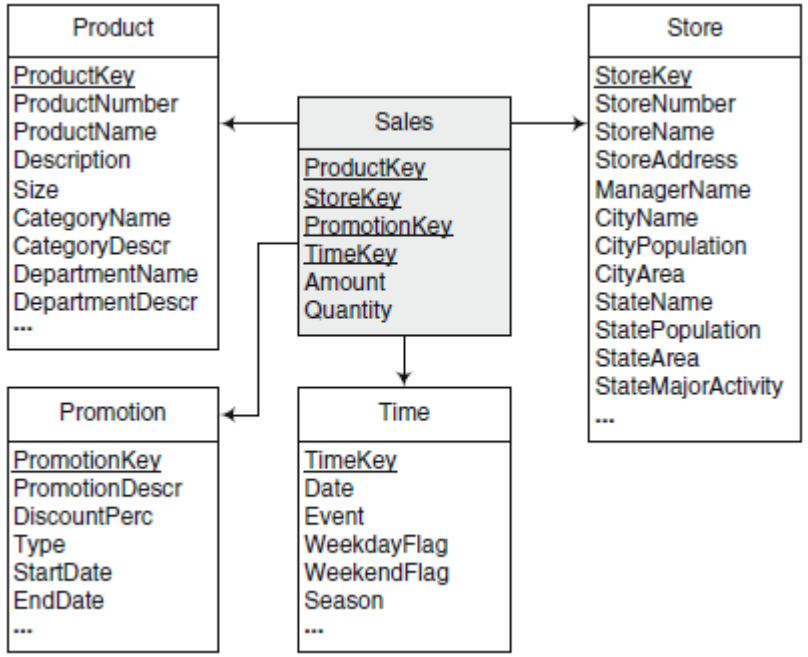


Quelle: A. Vaisman and E. Zimányi, Data Warehouse Systems, Data-Centric Systems and Applications, DOI 10.1007/978-3-642-54655-6, © Springer-Verlag Berlin Heidelberg 2014
Baars, Kemper, Business Intelligence & Analytics – Grundlagen und praktische Anwendungen, 4. Auflage, SpringerVieweg, 2021

Relational Data Warehouse Design

Star Schema

- Zentrale Faktentabelle (fact table) `Sales`
- Dimensionen `Product`, `Promotion`, `Time` und `Store`
- Die Faktentabelle hat als foreign keys die Schlüssel der Dimensions-Tabellen
- Die Faktentabelle ist normalisiert
- Die Dimensions-Tabelle sind i.d.R. nicht normalisiert, d.h. sie können redundante Daten enthalten.
- Beispiel für die Dimension `Product`: Alle Produkte der gleichen Kategorie haben redundante Informationen für die Attribute, die die Kategorie und das Department beschreiben.
- Beispiel in WideWorldImportersDW, Dimension.Supplier, (ORDER BY CATEGORY)



	Supplier Key	WWI Supplier ID	Supplier	Category	Primary Contact	Supplier Reference
1	7	4	Fabrikam, Inc.	Clothing Supplier	Bill Lawson	293092
2	17	4	Fabrikam, Inc.	Clothing Supplier	Bill Lawson	293092
3	3	3	Consolidated Messenger	Courier	Kerstin Pam	209340283
4	16	3	Consolidated Messenger	Courier	Kerstin Pam	209340283
5	27	3	Consolidated Messenger	Courier Services Supplier	Kerstin Pam	209340283
6	26	13	Woodgrove Bank	Financial Services Supplier	Hubert Helms	028034202

Quelle: A. Vaisman and E. Zimányi, Data Warehouse Systems, Data-Centric Systems and Applications, DOI 10.1007/978-3-642-54655-6, © Springer-Verlag Berlin Heidelberg 2014
Baars, Kemper, Business Intelligence & Analytics – Grundlagen und praktische Anwendungen, 4. Auflage, SpringerVieweg, 2021

Relational Data Warehouse Design

Star Schema

+ **Vorteile**
- Einfache und daher intuitive Datenmodelle
- Geringe Anzahl von Join-Operationen
- Geringe Anzahl physischer Data-Warehouse-Tabellen
- Geringer Aufwand im Rahmen der Data-Warehouse-Wartung

+ **Nachteile**
- Verschlechtertes Antwortzeitverhalten bei sehr großen Dimensions-Tabellen
- Redundanz innerhalb der Dimensions-Tabellen durch das mehrfache Festhalten identischer Fakten

Quelle: A. Vaisman and E. Zimányi, Data Warehouse Systems, Data-Centric Systems and Applications, DOI 10.1007/978-3-642-54655-6, © Springer-Verlag Berlin Heidelberg 2014
Baars, Kemper, Business Intelligence & Analytics – Grundlagen und praktische Anwendungen, 4. Auflage, SpringerVieweg, 2021

Relational Data Warehouse Design

Snowflake Schema

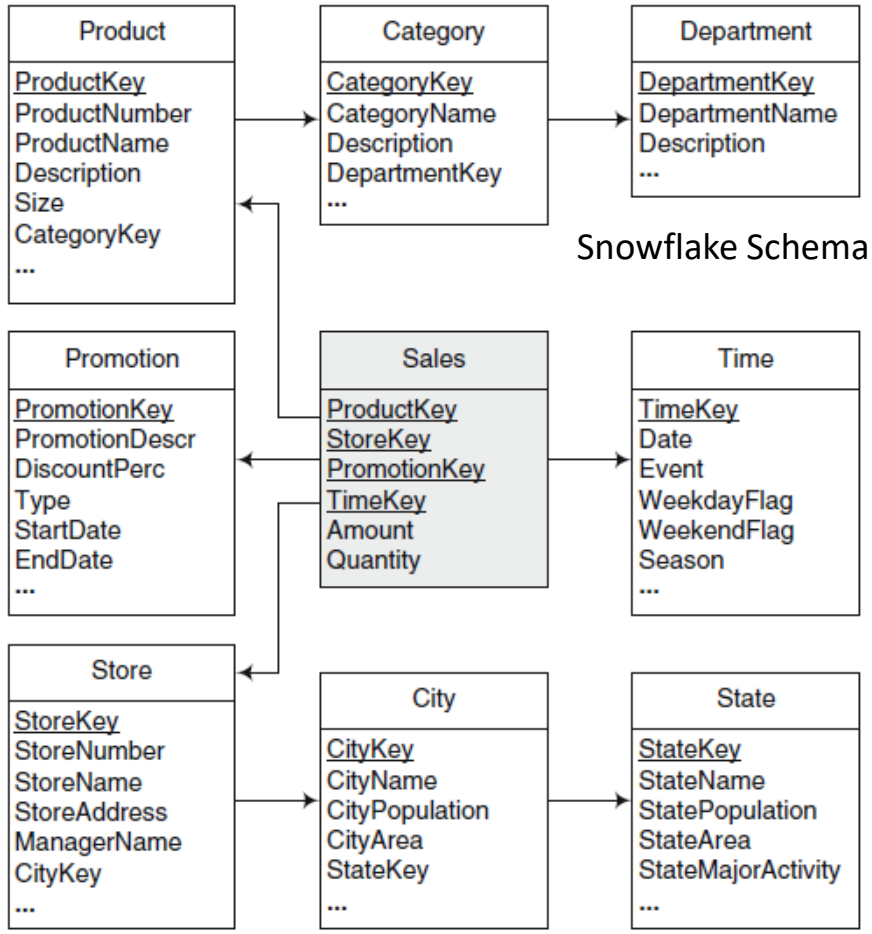
+ Normalisierung der Dimensions-Tabellen → Redundanzen werden vermieden
- Nach Baars ist eine vollständige Normalisierung nicht praktikabel
+ **Vorteile**
- normalisierte Tabellen optimieren den Speicherbedarf
- leichtere Wartbarkeit
+ **Nachteile**
- schlechtere Performance aufgrund vieler Joins

Starflake Schema

+ combination of the star and the snowflake schemas, where some dimensions are normalized while others are not

Constellation Schema

+ has multiple fact tables that share dimension tables



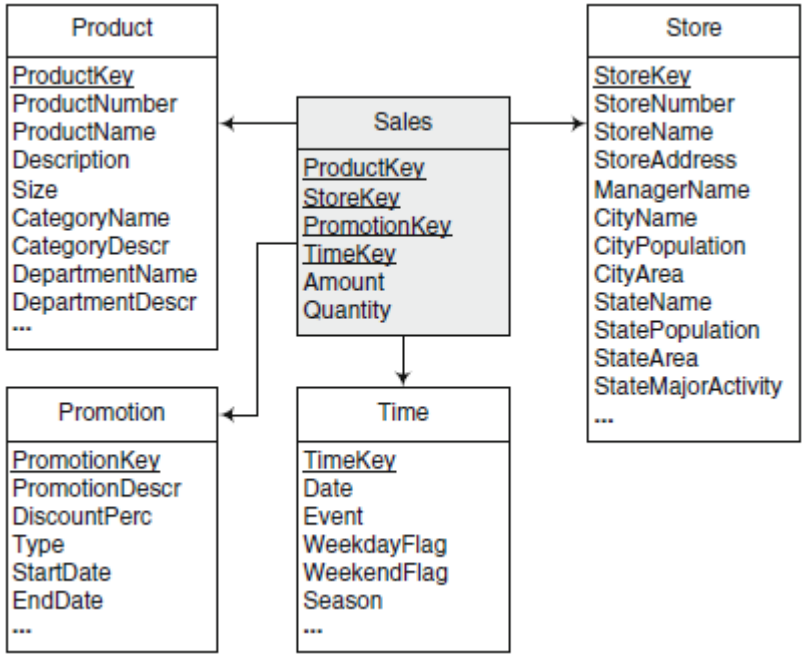
Snowflake Schema

Quelle: A. Vaisman and E. Zimányi, Data Warehouse Systems, Data-Centric Systems and Applications, DOI 10.1007/978-3-642-54655-6, © Springer-Verlag Berlin Heidelberg 2014
Baars, Kemper, Business Intelligence & Analytics – Grundlagen und praktische Anwendungen, 4. Auflage, SpringerVieweg, 2021

Relational Data Warehouse Design

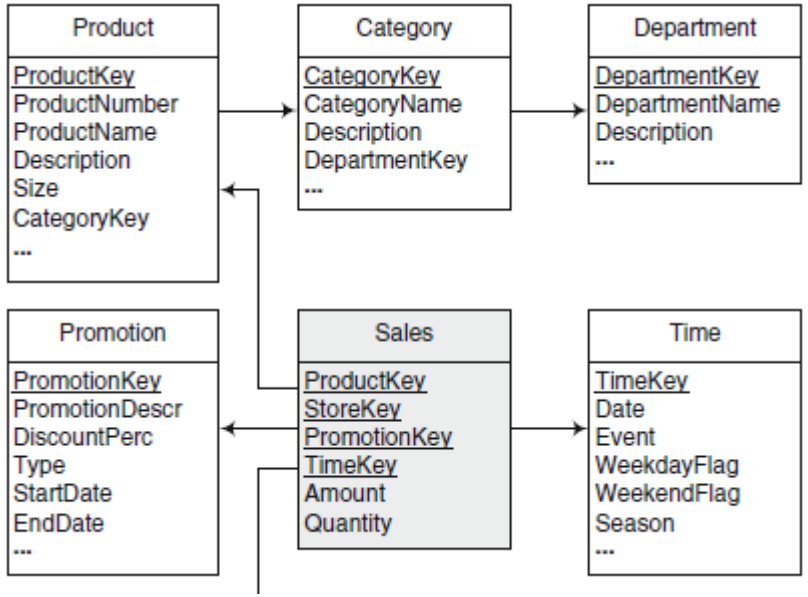
SQL-Abfrage Star Schema

```
SELECT CategoryName, SUM(Amount)
FROM Product P, Sales S
WHERE P.ProductKey = S.ProductKey
GROUP BY CategoryName
```



SQL-Abfrage Snowflake Schema

```
SELECT CategoryName, SUM(Amount)
FROM Product P, Category C, Sales S
WHERE P.ProductKey = S.ProductKey
          AND P.CategoryKey = C.CategoryKey
GROUP BY CategoryName
```



Quelle: A. Vaisman and E. Zimányi, Data Warehouse Systems, Data-Centric Systems and Applications, DOI 10.1007/978-3-642-54655-6, © Springer-Verlag Berlin Heidelberg 2014

Relational Data Warehouse Design

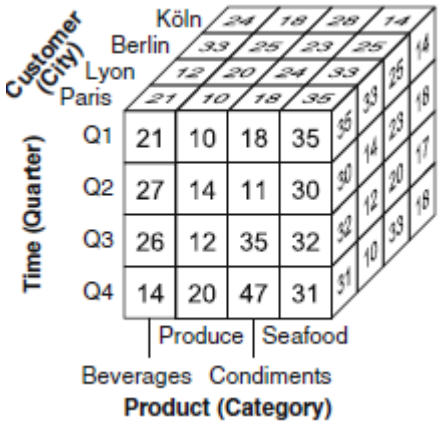
MultiDim-Modell → Relational Model

+ **Rule 1**: A level L, provided it is not related to a fact with a one-to-one relationship, is mapped to a table TL that contains all attributes of the level. A surrogate key may be added to the table; otherwise, the identifier of the level will be the key of the table. Note that additional attributes will be added to this table when mapping relationships using Rule 3 below.
+ **Rule 2**: A fact F is mapped to a table TF that includes as attributes all measures of the fact. Further, a surrogate key may be added to the table. Note that additional attributes will be added to this table when mapping relationships using Rule 3 below.
+ **Rule 3**: A relationship between either a fact F and a dimension level L, or between dimension levels LP and LC (standing for the parent and child levels, respectively), can be mapped in three different ways, depending on its cardinalities:

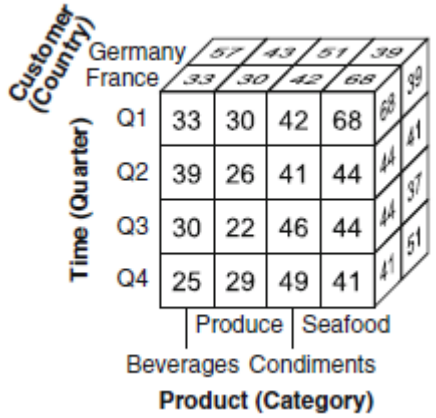
- **Rule 3a**: If the relationship is one-to-one, the table corresponding to the fact (TF) or to the child level (TC) is extended with all the attributes of the dimension level or the parent level, respectively.
- **Rule 3b**: If the relationship is one-to-many, the table corresponding to the fact (TF) or to the child level (TC) is extended with the surrogate key of the table corresponding to the dimension level (TL) or the parent level (TP), respectively, that is, there is a foreign key in the fact or child table pointing to the other table.
- **Rule 3c**: If the relationship is many-to-many, a new table TB (standing for bridge table) is created that contains as attributes the surrogate keys of the tables corresponding to the fact (TF) and the dimension level (TL), or the parent (TP) and child levels (TC), respectively. The key of the table is the combination of both surrogate keys. If the relationship has a distributing attribute, an additional attribute is added to the table to store this information.

Data Warehouse
OLAP Operations (1)

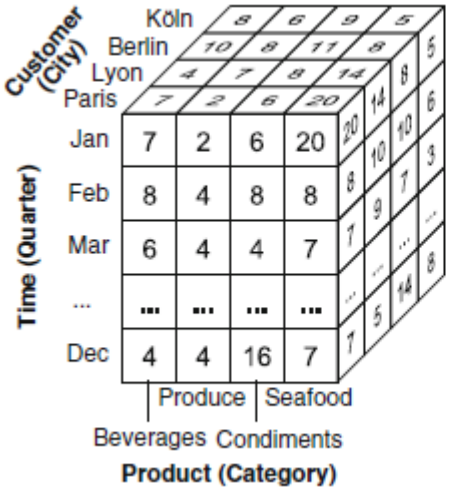
- Data Cube mit drei Dimensionen: Product, Time, Customer
- Aggregation: Product nach Kategorie, Time nach Quarter, Customer nach der Stadt



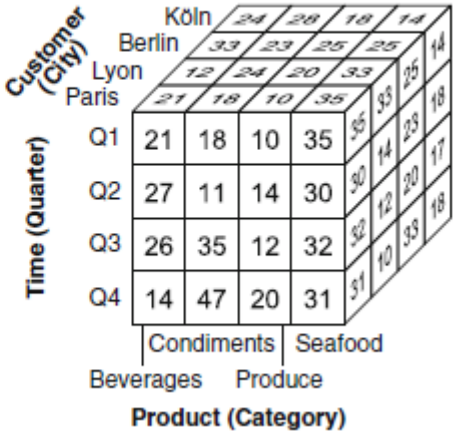
Original cube



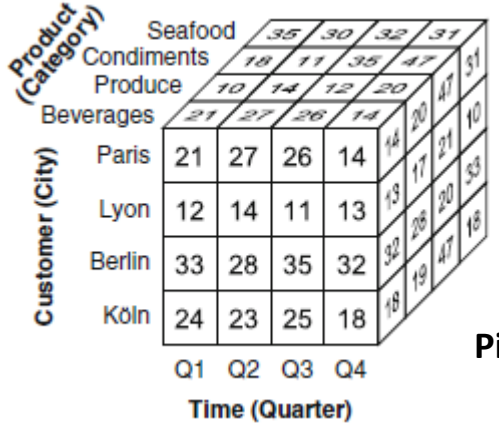
Roll-up to the Country level



Drill-down to the Month level



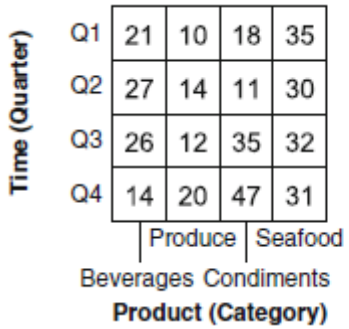
Sort product by name



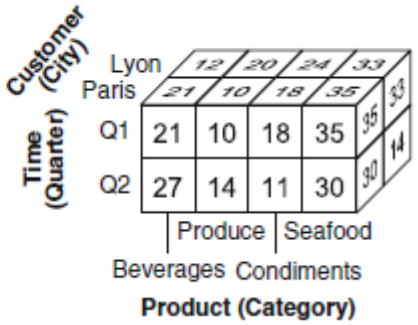
Pivot (Drehung des Cubes)

Quelle: A. Vaisman and E. Zimányi, Data Warehouse Systems, Data-Centric Systems and Applications, DOI 10.1007/978-3-642-54655-6, © Springer-Verlag Berlin Heidelberg 2014

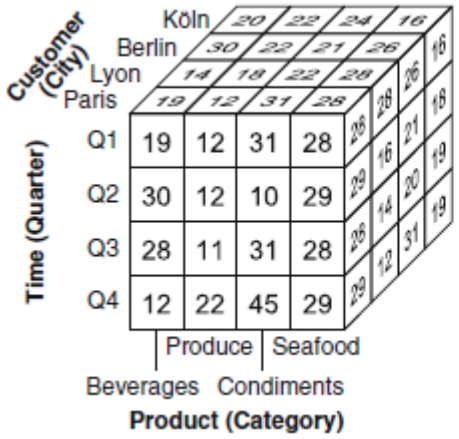
Data Warehouse
OLAP Operations (2)



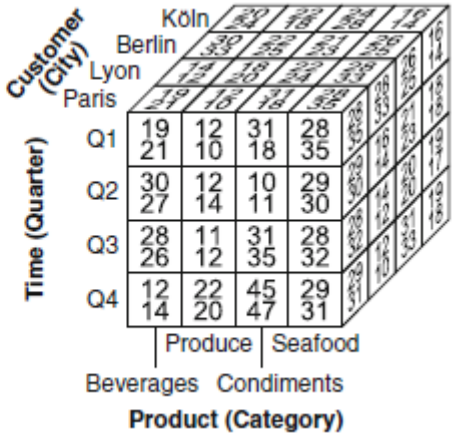
Slice on City='Paris'.



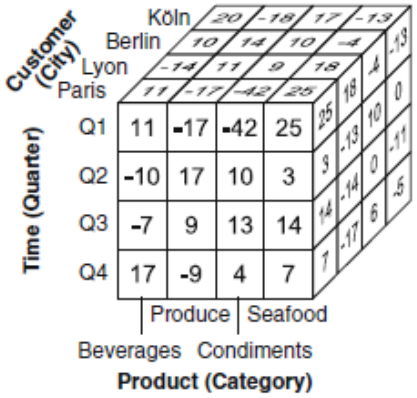
Dice on City='Paris' or 'Lyon' and Quarter='Q1' or 'Q2'



Cube for 2011

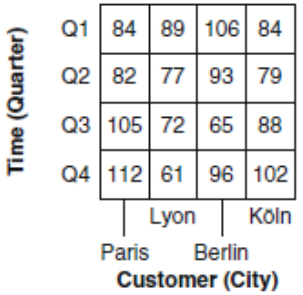


Drill-across operation

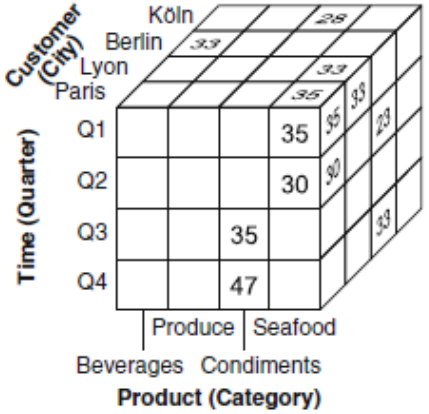


Änderung in Prozent

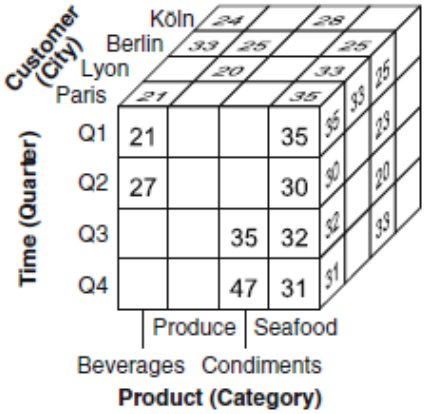
k



Total sales by quarter and city

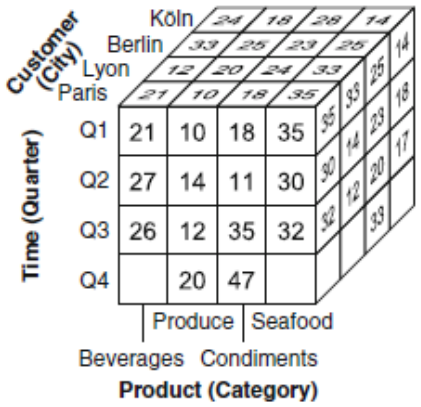


Maximum sales by quarter and city

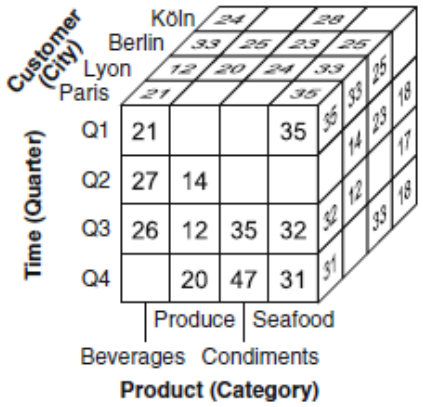


Top two sales by quarter and city

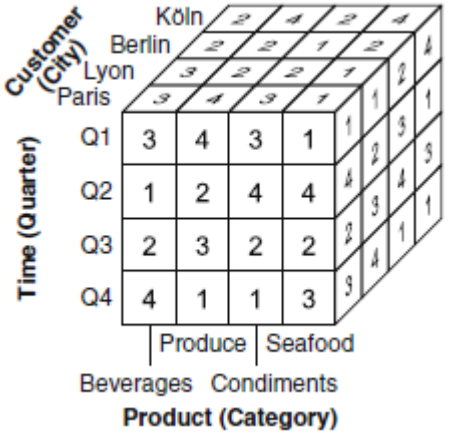
Data Warehouse
OLAP Operations (3)



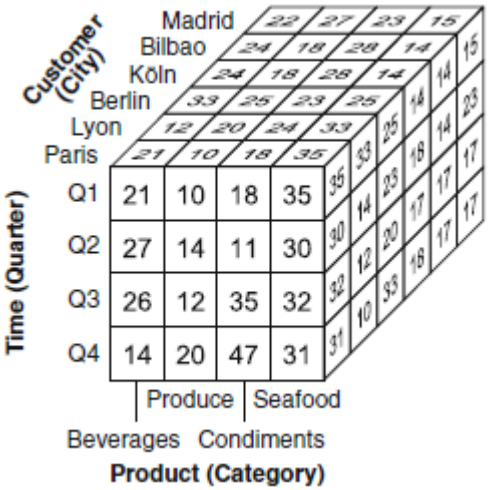
Top 70% by city and category ordered by ascending quarter



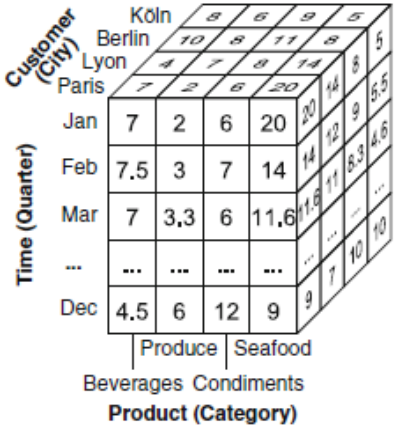
Top 70% by city and category ordered by descending quantity



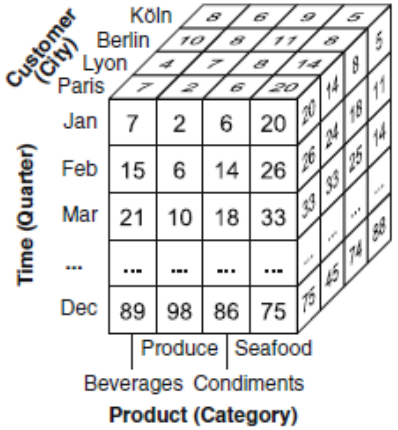
Rank quarter by category and city ordered by descending quantity



Union of the original cube and another cube with data from Spain



Three-month moving average



Year-to-date computation

Quelle: A. Vaisman and E. Zimányi, Data Warehouse Systems, Data-Centric Systems and Applications, DOI 10.1007/978-3-642-54655-6, © Springer-Verlag Berlin Heidelberg 2014

Data Warehouse
Data Cube Construction (1)

Zwei-dimensionaler Data Cube

+ Übung 28-Two-Dim-Data-Cube
+ Dimensionen `Product` und `Customer`
+ Measure `SalesAmount`

	c1	c2	c3	Total
p1	100	105	100	305
p2	70	60	40	170
p3	30	40	50	120
Total	200	205	190	595

+ Data Cube enthält alle Aggregationen der Cube Cells, also
- `SalesAmount` by `Product`,
- `SalesAmount` by `Customer`,
- `SalesAmount` by `Product` and `Costumer`

Fact table

+ Fact table für den zwei-dimensionalen Data Cube

ProductKey	CustomerKey	SalesAmount
p1	c1	100
p1	c2	105
p1	c3	100
p2	c1	70
p2	c2	60
p2	c3	40
p3	c1	30
p3	c2	40
p3	c3	50

Data Warehouse
Data Cube Construction (2)

SQL-Befehl zur Erstellung der `Product`-Tabelle

```
CREATE TABLE [Product](
[ProductKey] [int] IDENTITY(1,1) NOT NULL,
[Name] [nvarchar](50) NOT NULL,
CONSTRAINT [PK_Dimension_Product_Item] PRIMARY KEY
CLUSTERED
([ProductKey] ASC))
```

SQL-Befehl zur Erstellung der `Customer`-Tabelle

```
CREATE TABLE [Customer](
[CustomerKey] [int] IDENTITY(1,1) NOT NULL,
[Customer] [nvarchar](50) NOT NULL,
CONSTRAINT [PK_Dimension_Customer] PRIMARY KEY
CLUSTERED
([CustomerKey] ASC))
```

Data Warehouse
Data Cube Construction (3)

SQL-Befehl zur Erstellung der `Fact`-Tabelle

+ Erstellung der SQL-Tabelle mit ForeignKey-Abhängigkeiten

```
CREATE TABLE [SalesAmountFact](
[SalesAmountKey][bigint] IDENTITY(1,1) NOT NULL,
[ProductKey][int] NOT NULL,
[CustomerKey][int] NOT NULL,
[SalesAmount][int] NOT NULL,
CONSTRAINT [SalesAmountKey] PRIMARY KEY CLUSTERED
([SalesAmountKey] ASC))
```

```
ALTER TABLE [SalesAmountFact]  WITH CHECK ADD
CONSTRAINT
[FK_SalesAmountFact_To_Customer_Key_Dimension_Custo
mer] FOREIGN KEY([CustomerKey])
REFERENCES [Customer] ([CustomerKey])

ALTER TABLE [SalesAmountFact] CHECK CONSTRAINT
[FK_SalesAmountFact_To_Customer_Key_Dimension_Custo
mer]
```

```
ALTER TABLE [SalesAmountFact] WITH CHECK ADD
CONSTRAINT
[FK_SalesAmountFact_To_Product_Key_Dimension_Produc
t] FOREIGN KEY([ProductKey])
REFERENCES [Product] ([ProductKey])

ALTER TABLE [SalesAmountFact] CHECK CONSTRAINT
[FK_SalesAmountFact_To_Product_Key_Dimension_Produc
t]
```

Data Warehouse
Data Cube Construction (4)

Datenbankstruktur

- dbo.Customer
  - Spalten
    - CustomerKey (PS, int, Nicht NULL)
    - Customer (nvarchar(50), Nicht NULL)

	CustomerKey	Customer
1	1	Jessica
2	2	John
3	3	Jennifer

- dbo.Product
  - Spalten
    - ProductKey (PS, int, Nicht NULL)
    - Name (nvarchar(50), Nicht NULL)

	ProductKey	Name
1	1	Fernseher
2	2	Handy
3	3	Tablet

- dbo.SalesAmountFact
  - Spalten
    - SalesAmountKey (PS, bigint, Nicht NULL)
    - ProductKey (FS, int, Nicht NULL)
    - CustomerKey (FS, int, Nicht NULL)
    - SalesAmount (int, Nicht NULL)

	SalesAmountKey	ProductKey	CustomerKey	SalesAmount
1	1	1	1	100
2	2	1	2	105
3	3	1	3	100
4	4	2	1	70
5	5	2	2	60
6	6	2	3	40
7	7	3	1	30
8	8	3	2	40
9	9	3	3	50

Quelle: A. Vaisman and E. Zimányi, Data Warehouse Systems, Data-Centric Systems and Applications, DOI 10.1007/978-3-642-54655-6, © Springer-Verlag Berlin Heidelberg 2014

Data Warehouse
Time Dimension

Time Dimension in OLTP-Datenbanken

- Üblicherweise als DATE (oder DATETIME)-Format
- Feiertage, Wochentage werden i.d.R. on-the-fly berechnet

Time Dimension in OLAP-Datenbanken

- Feiertage, Wochenende, Wochentage werden i.d.R. in einer Spalte als “Flag” gekennzeichnet

```
SELECT SUM(SalesAmount)
FROM Time T, Sales S
WHERE T.TimeKey = S.TimeKey AND T.WeekendFlag
```

Data Warehouse
Many-to-Many Dimensions

Many-to-Many Dimensions

- \+ Nutzung einer zusätzlichen Bridge-Tabelle zwischen der Faktentabelle und der Dimension.
- \+ Bridge-Tabelle BalanceClient + surrogate key (BalanceKey)

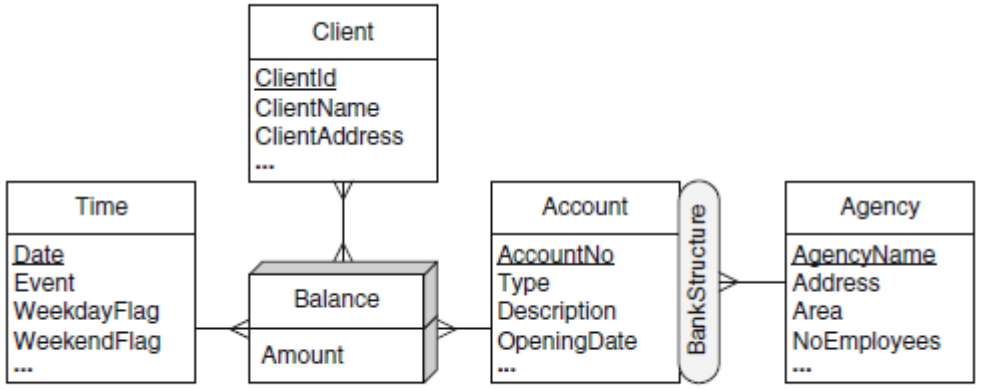


Fig. 4.19 Multidimensional schema for the analysis of bank accounts

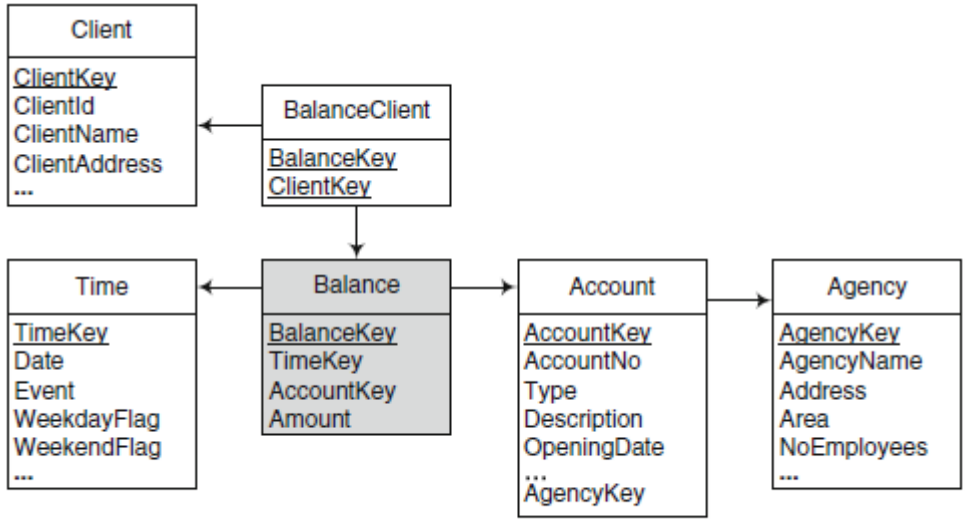


Fig. 5.14 Relations for the many-to-many dimension in Fig. 4.19

Data Warehouse
Slowly Changing Dimensions (1)

Bisherige Annahme

+ Neue Daten gehören zu bestehenden Facts
+ Dimensionen sind stabil, Daten der Dimensions-Tabellen ändern sich nicht
+ Die Werte (Tuples) des `Sales` fact table werden geändert, wenn Produkte verkauft werden.

Slowly Changing Dimensions

+ Dimensionen können sich ändern bezüglich
- der Struktur und
- der Instanzen
+ Beispiel
- `Sales` fact (Star-Schema) mit den Dimensionen `Time`, `Employee`, `Customer` und `Product`, Measure `SalesAmount`

+ Gründe für eine Änderung der Dimension
- Neues Produkt → Anpassung des `Product` tables
- Korrektur des `Product` tables aufgrund fehlerhafter Einträge
- Neue Kategorie (z.B. Änderung der Business-Strategie)

TimeKey	EmployeeKey	CustomerKey	ProductKey	SalesAmount
t1	e1	c1	p1	100
t2	e2	c2	p1	100
t3	e1	c3	p3	100
t4	e2	c4	p4	100

`Sales` fact table

ProductKey	ProductName	Discontinued	CategoryName	Description
p1	prod1	No	cat1	desc1
p2	prod2	No	cat1	desc1
p3	prod3	No	cat2	desc2
p4	prod4	No	cat2	desc2

`Product` dimension table

Data Warehouse
Slowly Changing Dimensions (2)

Fortsetzung des Beispiels

- Abfrage der Verkaufszahlen pro Arbeitnehmer und Produkt-Kategorie:

```
SELECT E.EmployeeKey, P.CategoryName,
SUM(SalesAmount)
FROM Sales S, Product P
WHERE S.ProductKey = P.ProductKey
GROUP BY E.EmployeeKey, P.CategoryName
```

- Ergebnis der Abfrage

EmployeeKey	CategoryName	SalesAmount
e1	cat1	100
e2	cat1	100
e1	cat2	100
e2	cat2	100

- Änderung in der Datenbank: Änderung der Klassifizierung des Produkts `prod1` zur Kategorie `cat2`.
- Das Abfrageergebnis würde sich nun ändern.

TimeKey	EmployeeKey	CustomerKey	ProductKey	SalesAmount
t1	e1	c1	p1	100
t2	e2	c2	p1	100
t3	e1	c3	p3	100
t4	e2	c4	p4	100

`Sales` fact table

ProductKey	ProductName	Discontinued	CategoryName	Description
p1	prod1	No	cat1	desc1
p2	prod2	No	cat1	desc1
p3	prod3	No	cat2	desc2
p4	prod4	No	cat2	desc2

`Product` dimension table

Data Warehouse
Slowly Changing Dimensions (3)

Behandlung von Slowly Changing Dimensions

+ Typ 1: Überschreibung des alten Attributwerts mit einem neuen Attributwert
- Nachteil: Verlust der Historie
- Anwendung, wenn die Änderung aufgrund eines fehlerhaften Eintrags in der Dimensions-Tabelle erfolgt.

Data Warehouse
Slowly Changing Dimensions (4)

Behandlung von Slowly Changing Dimensions

+ Typ 2: Versionierung der Tupel in der Dimensions-Tabelle
- Nachteil: schlechtere Performance bei Abfragen (z.B. Verkaufszahlen des Produkts `prod1`)
- Im Beispiel wird in der Dimensions-Tabelle `Product` für das Produkt `prod1` eine neue Zeile mit der neuen Kategorie `cat2` ergänzt. Zusätzlich werden zwei Spalten für die Zeitspanne der Gültigkeit ergänzt.
- Mögliche Ergänzung um eine Spalte (Flag), die den Status der Zeile erläutert.

Product Key	Product Name	Discontinued	Category Name	Description	From	To	Row Status
p1	prod1	No	cat1	desc1	2010-01-01	2011-12-31	Expired
p11	prod1	No	**cat2**	desc2	2012-01-01	9999-12-31	Current
...	...	...	...	...	...	...	...

`Product` dimension table mit Status-Flag

ProductKey	ProductName	Discontinued	CategoryName	Description
p1	prod1	No	cat1	desc1
p2	prod2	No	cat1	desc1
p3	prod3	No	cat2	desc2
p4	prod4	No	cat2	desc2

Ursprünglicher `Product` dimension table (Star-Schema)

Product Key	Product Name	Discontinued	Category Name	Description	From	To
p1	prod1	No	cat1	desc1	2010-01-01	2011-12-31
p11	prod1	No	cat2	desc2	2012-01-01	9999-12-31
p2	prod2	No	cat1	desc1	2012-01-01	9999-12-31
p3	prod3	No	cat2	desc2	2012-01-01	9999-12-31
p4	prod4	No	cat2	desc2	2012-01-01	9999-12-31

angepasster `Product` dimension table

Data Warehouse
Slowly Changing Dimensions (5)

Behandlung von Slowly Changing Dimensions

+ Typ 3: Einfügen einer Spalte für jede Änderung (attribute addition)
- Nachteil: Fehlende Historie (Gültigkeit)
- Im Beispiel wird bei einer Änderung der Kategorie eine neue Spalte für die Kategorie (`New Category`) und für die Beschreibung (`New Description`) eingefügt.

ProductKey	ProductName	Discontinued	CategoryName	Description
p1	prod1	No	cat1	desc1
p2	prod2	No	cat1	desc1
p3	prod3	No	cat2	desc2
p4	prod4	No	cat2	desc2

Ursprünglicher `Product` dimension table (Star-Schema)

Product Key	Product Name	Discontinued	Category Name	New Category	Description	New Description
p1	prod1	No	**cat1**	**cat2**	**desc1**	**desc2**
p2	prod2	No	cat1	Null	desc1	Null
p3	prod3	No	cat2	Null	desc2	Null
p4	prod4	No	cat2	Null	desc2	Null

`Product` dimension table mit neuer Spalte bei Änderung der Kategorie

Quelle: A. Vaisman and E. Zimányi, Data Warehouse Systems, Data-Centric Systems and Applications, DOI 10.1007/978-3-642-54655-6, © Springer-Verlag Berlin Heidelberg 2014

Data Warehouse
Slowly Changing Dimensions (6)

Behandlung von Slowly Changing Dimensions

+ Typ 4 und Typ 5: Einführung einer Tabelle `minidimension` für Attribute, die sich häufig ändern
- Nachteil: Fehlende Historie (Zeitraum der Gültigkeit)
- Im Beispiel wird bei einer Änderung der Kategorie eine neue Spalte für die Kategorie (`New Category`) und für die Beschreibung (`New Description`) eingefügt.

Quelle: A. Vaisman and E. Zimányi, Data Warehouse Systems, Data-Centric Systems and Applications, DOI 10.1007/978-3-642-54655-6, © Springer-Verlag Berlin Heidelberg 2014

Data Warehouse
Data Cube Abfrage (1)

SQL-Befehl zur Abfrage des zwei-dimensionalen Facts

+ Rollup-Operator
- group subtotals in the order given by a list of attributes

```
SELECT ProductKey, CustomerKey, SUM(SalesAmount) as
SumSalesAmount
FROM SalesAmountFact
GROUP BY ROLLUP(ProductKey, CustomerKey)
```

	ProductKey	CustomerKey	SumSalesAmount
1	1	1	100
2	1	2	105
3	1	3	100
4	1	NULL	305
5	2	1	70
6	2	2	60
7	2	3	40
8	2	NULL	170
9	3	1	30
10	3	2	40
11	3	3	50
12	3	NULL	120
13	NULL	NULL	595

+ Cube-Operator
- computes all totals of such a list

```
SELECT ProductKey, CustomerKey, SUM(SalesAmount) as
SumSalesAmount
FROM SalesAmountFact
GROUP BY CUBE(ProductKey, CustomerKey)
```

	ProductKey	CustomerKey	SumSalesAmount
1	1	1	100
2	2	1	70
3	3	1	30
4	NULL	1	200
5	1	2	105
6	2	2	60
7	3	2	40
8	NULL	2	205
9	1	3	100
10	2	3	40
11	3	3	50
12	NULL	3	190
13	NULL	NULL	595
14	1	NULL	305
15	2	NULL	170
16	3	NULL	120

Data Warehouse
Data Cube Abfrage (2)

SQL-Befehl zur Abfrage des zwei-dimensionalen Facts

+ GROUPING SETS
- Der ROLLUP-Befehl

```
SELECT ProductKey, CustomerKey, SUM(SalesAmount)
FROM SalesAmountFact
GROUP BY ROLLUP(ProductKey, CustomerKey)
```

- kann auch geschrieben werden als

```
SELECT ProductKey, CustomerKey, SUM(SalesAmount)
FROM SalesAmountFact
GROUP BY GROUPING SETS((ProductKey, CustomerKey),
(ProductKey), ())
```

Data Warehouse
Data Cube Abfrage – Window Functions (1)

Window Functions

+ Vergleich detaillierter Daten mit aggregierten Werten

Window Partitioning

+ Beispiel: Relevanz der Verkaufszahlen zu verschiedenen Kunden

```
SELECT ProductKey, CustomerKey, SalesAmount,
MAX(SalesAmount) OVER
(PARTITION BY ProductKey) AS MaxAmount
FROM SalesAmountFact
```

+ Ergebnis: Verkaufszahlen von Produkten an Kunden im Vergleich zu der maximalen Verkaufszahl (an einen Kunden) für das jeweilige Produkt
+ Die ersten drei Spalten bilden den Inhalt der SalesAmountFact-Tabelle ab.
+ Die vierte Spalte (`MaxAmount`) sucht über das Fenster (Window, Partition) `ProductKey` den maximalen Wert von `SalesAmount`. Das Fenster fasst Zeilen mit gleichem `ProductKey` zusammen.

	ProductKey	CustomerKey	SalesAmount	MaxAmount
1	1	1	100	105
2	1	2	105	105
3	1	3	100	105
4	2	1	70	70
5	2	2	60	70
6	2	3	40	70
7	3	1	30	50
8	3	2	40	50
9	3	3	50	50

Data Warehouse
Data Cube Abfrage – Window Functions (2)

Window Ordering

- Sortierung der Zeilen in einer Partition
- Nützlich zur Berechnung einer Rangfolge (Ranking)
- Beispiel: Reihenfolge der Verkaufszahlen eines Produkts für jeden Kunden

```
SELECT ProductKey, CustomerKey, SalesAmount,
ROW_NUMBER() OVER
(PARTITION BY CustomerKey ORDER BY
SalesAmount DESC) AS [Row#]
FROM SalesAmountFact
```

- Ergebnis: Die Zeilen werden nach dem `CustomerKey` partitioniert und nach dem `SalesAmount` sortiert.

	ProductKey	CustomerKey	SalesAmount	Row#
1	1	1	100	1
2	2	1	70	2
3	3	1	30	3
4	1	2	105	1
5	2	2	60	2
6	3	2	40	3
7	1	3	100	1
8	3	3	50	2
9	2	3	40	3

Data Warehouse
Data Cube Abfrage – Window Functions (3)

Window Ordering w/ RANK

- Bildung eines Rankings in einer Partition
- Beispiel: Partitionierung über das Produkt und Berechnung der Rangfolge (Ranking) der Kunden über die Verkaufszahl

```
SELECT ProductKey, CustomerKey, SalesAmount, RANK()
OVER
(PARTITION BY ProductKey ORDER BY SalesAmount DESC)
AS [Rank]
FROM SalesAmountFact
```

	ProductKey	CustomerKey	SalesAmount	Rank
1	1	2	105	1
2	1	3	100	2
3	1	1	100	2
4	2	1	70	1
5	2	2	60	2
6	2	3	40	3
7	3	3	50	1
8	3	2	40	2
9	3	1	30	3

Data Warehouse
Data Cube Abfrage – Window Functions (4)

Window Framing

- Definition der Größe einer Partition zur Berechnung statistischer Größen
- Beispiel 1: Berechnung des Moving Average über drei Monate für die Verkaufszahlen pro Produkt

```
SELECT ProductKey, [Year], [Month], SalesAmount,
AVG(SalesAmount) OVER
(PARTITION BY ProductKey ORDER BY [Year], [Month]
ROWS 2 PRECEDING) AS MovAvg
FROM SalesAmountFact
```

Product Key	Year	Month	Sales Amount	MovAvg
p1	2011	10	100	100
p1	2011	11	105	102.5
p1	2011	12	100	101.67
p2	2011	12	60	60
p2	2012	1	40	50
p2	2012	2	70	56.67
p3	2012	1	30	30
p3	2012	2	50	40
p3	2012	3	40	40

- Beispiel 2: Kumulierte Summe der Verkaufszahlen pro Produkt im jeweiligen Jahr (year-to-date (YTD) sum)

```
SELECT ProductKey, [Year], [Month], SalesAmount,
SUM(SalesAmount) OVER
(PARTITION BY ProductKey, [Year] ORDER BY [Month]
ROWS UNBOUNDED PRECEDING) AS YTD
FROM SalesAmountFact
```

Product Key	Year	Month	Sales Amount	YTD
p1	2011	10	100	100
p1	2011	11	105	205
p1	2011	12	100	305
p2	2011	12	60	60
p2	2012	1	40	40
p2	2012	2	70	110
p3	2012	1	30	30
p3	2012	2	50	80
p3	2012	3	40	120

```
SELECT ProductKey, [Year], [Month], SalesAmount, SUM(SalesAmount) AS YTD
FROM SalesAmountFact S1, SalesAmountFact S2
WHERE S1.ProductKey = S2.ProductKey AND
S1.[Year] = S2.[Year] AND S1.[Month] >= S2.[Month]
```

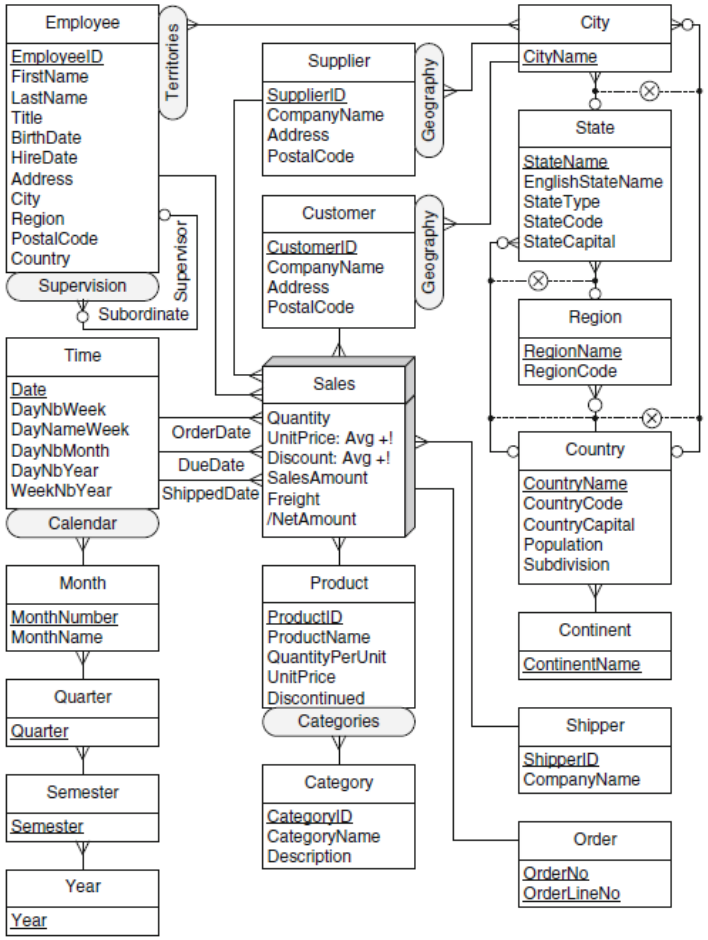
Quelle: A. Vaisman and E. Zimányi, Data Warehouse Systems, Data-Centric Systems and Applications, DOI 10.1007/978-3-642-54655-6, © Springer-Verlag Berlin Heidelberg 2014

Data Warehouse
Übungsaufgabe 14 (1/2)

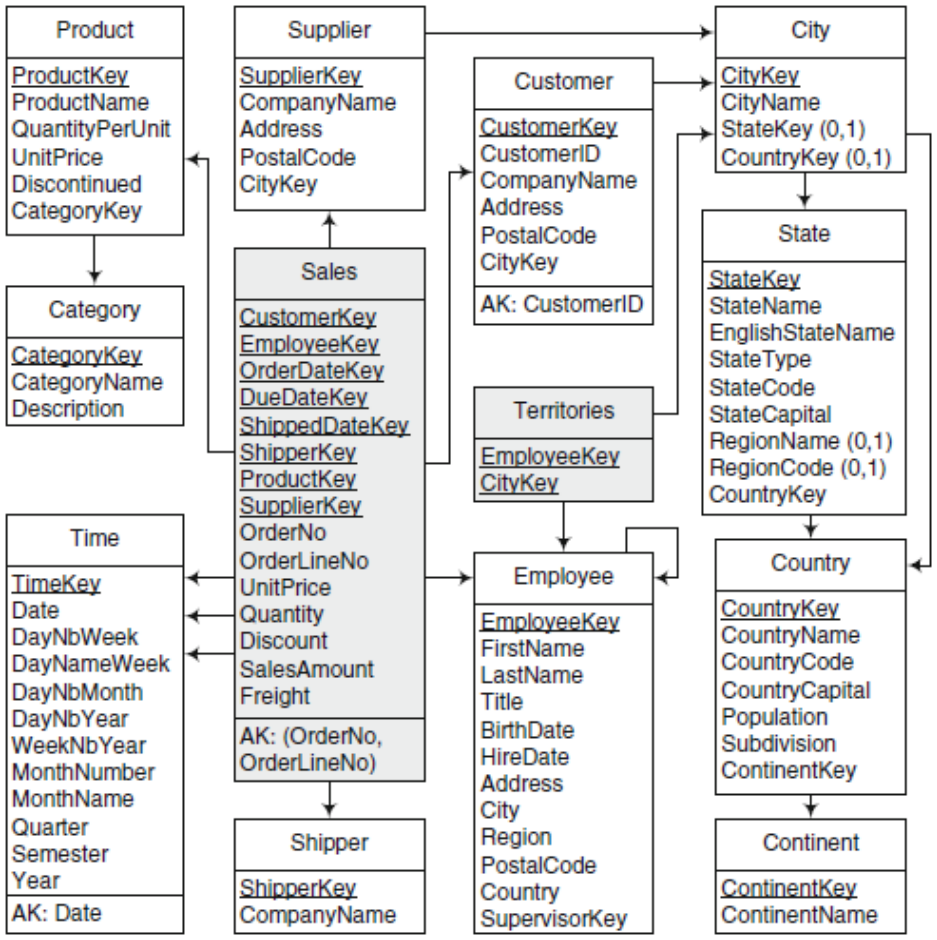
MultiDim-Modell → Relational Model

- Auf der nächsten Seite ist ein Data Warehouse als MultiDim-Modell (conceptual schema) und die Umsetzung in einer relationalen Datenbank (relational representation) dargestellt.
- Geben Sie Beispiele (sofern vorhanden) für die Anwendung der Regeln zur Umwandlung des MultiDim-Modells in das relationale Modell (siehe Folie 15) an.
- Erstellen Sie eine kurze Präsentation, in der Sie die Aufgabenstellung und Ihre Beispiele festhalten.

Data Warehouse
Übungsaufgabe 14 (2/2)



Conceptual schema of the Northwind data warehouse



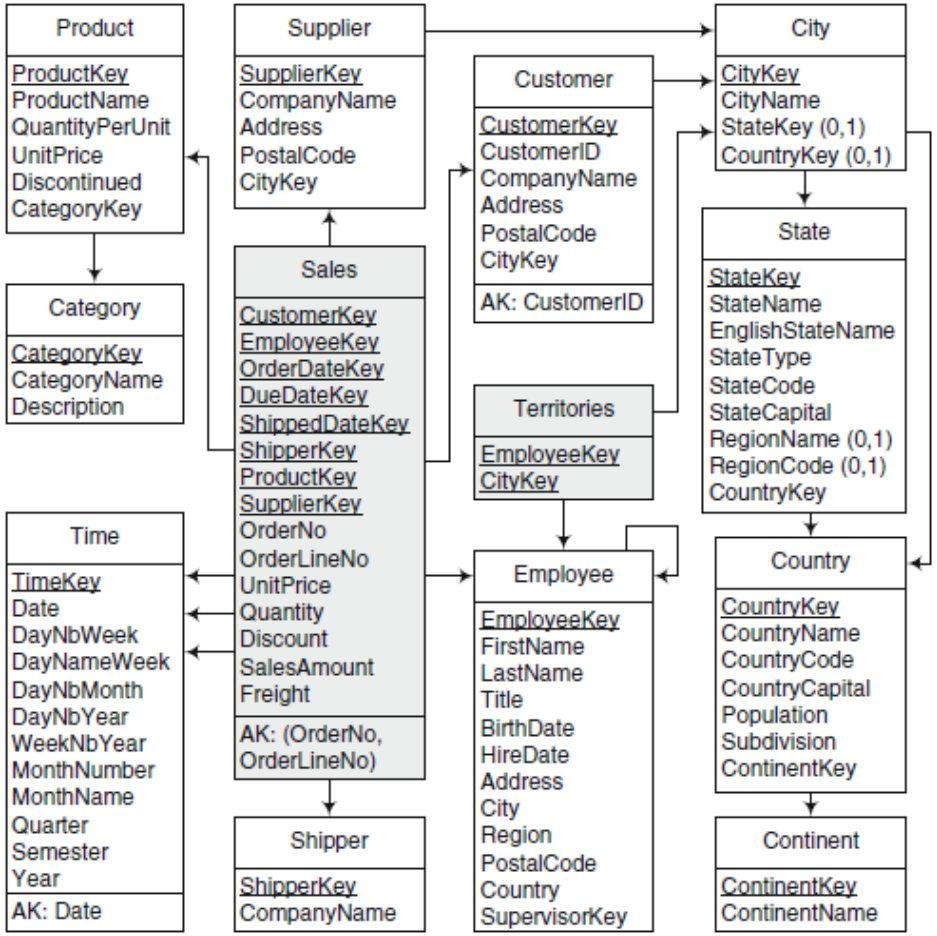
Relational representation of the Northwind data warehouse

Quelle: A. Vaisman and E. Zimányi, Data Warehouse Systems, Data-Centric Systems and Applications, DOI 10.1007/978-3-642-54655-6, © Springer-Verlag Berlin Heidelberg 2014

Data Warehouse Übungsaufgabe 15

SQL für ein relationales Data Warehouse-Modell

- Schreiben Sie für das abgebildete relationale Modell des Northwind Data Warehouses SQL-Statements zur Erstellung der SQL-Tabellen. Zur Vereinfachung können Sie die `alternate keys` (AK) weglassen.
- Erstellen Sie eine kurze Präsentation, in der Sie die Aufgabenstellung und die Vorgehensweise erläutern.
- Hinweis: `Sales` und `Territories` sind Fact-Tabellen



Relational representation of the Northwind data warehouse

Quelle: A. Vaisman and E. Zimányi, Data Warehouse Systems, Data-Centric Systems and Applications, DOI 10.1007/978-3-642-54655-6, © Springer-Verlag Berlin Heidelberg 2014

Hochschule Karlsruhe
University of
Applied Sciences

www.h-ka.de