Daimler TSS

A company of Daimler AG

LECTURE @DHBW: DATA WAREHOUSE

PART II: DWH DATA MODELING & OLAP

ANDREAS BUCKENHOFER, DAIMLER TSS

DHBW
Duale Hochschule
Baden-Württemberg

ABOUT ME

Andreas Buckenhofer
Senior DB Professional
andreas.buckenhofer@daimler.com

Since 2009 at Daimler TSS
Department: Big Data
Business Unit: Analytics

- https://de.linkedin.com/in/buckenhofer
- https://twitter.com/ABuckenhofer
- DOAG https://www.doag.org/de/themen/datenbank/in-memory/
- DHBW http://wwwlehre.dhbw-stuttgart.de/~buckenhofer/
- XING https://www.xing.com/profile/Andreas_Buckenhofer2

1995 – Datenbanken
- Admin
- Tuning
- Entwicklung
- Modellierung

1998 – Oracle DB
- ab V7.3.4

1998 – DWH / CRM
- Architektur
- Integration
- Reporting

2004 – RZ Migration
- 2 Jahre UK

2006 – Siebel CRM
- ab V7.8

2006 – Replication / CDC
- Oracle CDC

2007 – SQL Server
- ab V2005
- SSIS

2009 – Data Vault

2010 – Cognos
- ab V8.4

2012 – DB2
- ab V9.7

2013 – Hadoop
- Architektur
- Datenintegration
- SQL-on-Hadoop

2015 – IIDR
- 10.2.1

NOT JUST AVERAGE: OUTSTANDING.

As a 100% Daimler subsidiary, we give 100 percent, always and never less. We love IT and pull out all the stops to aid Daimler's development with our expertise on its journey into the future.

Our objective: We make Daimler the most innovative and digital mobility company.

INTERNAL IT PARTNER FOR DAIMLER

+ Holistic solutions according to the Daimler guidelines
    + IT strategy
    + Security
    + Architecture
+ Developing and securing know-how
+ TSS is a partner who can be trusted with sensitive data

As subsidiary: **maximum added value** for Daimler

+ Market closeness
+ Independence
+ Flexibility (short decision making process, ability to react quickly)

LOCATIONS

Daimler TSS Germany

7 locations

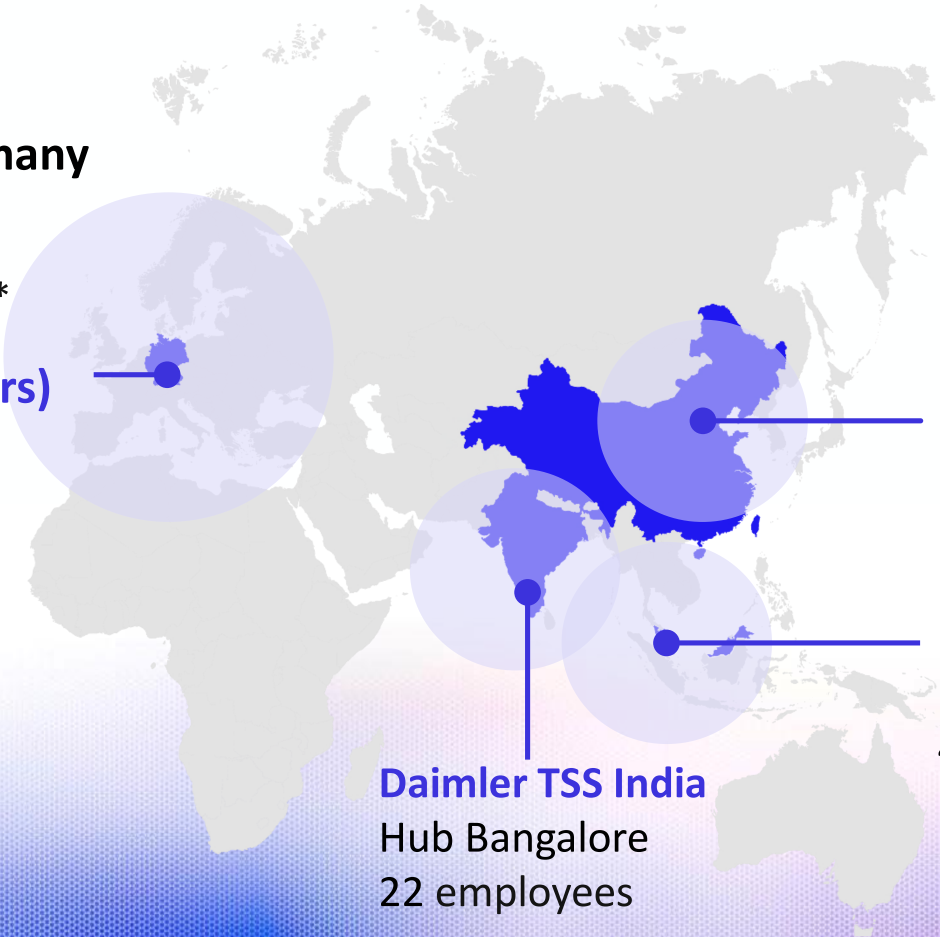
1000 employees*

Ulm (Headquarters)

Stuttgart

Berlin

Karlsruhe



Daimler TSS China

Hub Beijing

10 employees

Daimler TSS Malaysia

Hub Kuala Lumpur

42 employees

Daimler TSS India

Hub Bangalore

22 employees

* as of August 2017

WHAT YOU WILL LEARN TODAY

After the end of this lecture you will be able to

Understand differences in data modeling between OLTP and OLAP

Understand why data modeling is important

Understand data modeling in the Core Warehouse Layer and Data Mart Layer

- Data Vault
- Dimensional Model / Star schema

Understand dimensions and facts

Understand ROLAP & MOLAP

DATA MODELING FOR OLTP APPLICATIONS

Requirements

- Efficient update and delete operations
- Efficient read operations
- Avoid contradiction in the data – don’t store data twice or multiple times
- Easy maintenance of the data model

→As little redundancy as possible in the data model

CODD‘S NORMAL FORMS FOR DB RELATIONS: 1NF

First Normal Form (1NF):

- A relation/table is in first normal form if
    - the domain of each attribute contains only **atomic** (simple, indivisible) values.
    - the value of any attribute in a tuple/row must be a single value from the domain of that attribute, i.e. **no attribute values can be sets**

CODD'S NORMAL FORMS FOR DB RELATIONS: 1NF

CD_ID	Album	Founded	Titels
11	Anastacia – Not that kind	1999	1. Not that kind, 2. I'm outta love, 3 Cowboys & Kisses
12	Pink Floyd – Wish you were here	1964	1. Shine on you crazy diamond
13	Anastacia – Freak of Nature	1999	1. Paid my dues

↓

CD_ID	Album	Performer	Founded	Track	Titels
11	Not that kind	Anastacia	1999	1	Not that kind
11	Not that kind	Anastacia	1999	2	I'm outta love
11	Not that kind	Anastacia	1999	3	Cowboys & Kisses
12	Wish you were here	Pink Floyd	1964	1	Shine on you crazy diamond
13	Freak of Nature	Anastacia	1999	1	Paid my dues

CODD‘S NORMAL FORMS FOR DB RELATIONS: 2NF

Second Normal Form (2NF):

- In 1st normal form
- Every non-key attribute is **fully dependent on the key**. There are no dependencies between a partial key and a non-key field.

CODD‘S NORMAL FORMS FOR DB RELATIONS: 2NF

CD_ID	Album	Performer	Founded	Track	Titels
11	Not that kind	Anastacia	1999	1	Not that kind
11	Not that kind	Anastacia	1999	2	I‘m outta love
11	Not that kind	Anastacia	1999	3	Cowboys & Kisses
12	Wish you were here	Pink Floyd	1964	1	Shine on you crazy diamond
13	Freak of Nature	Anastacia	1999	[illegible]	[illegible]

CD_ID	Album	Performer	Founded
11	Not that kind	Anastacia	1999
12	Wish you were here	Pink Floyd	1964
13	Freak of Nature	Anastacia	1999

CD_ID	Track	Titels
11	1	Not that kind
11	2	I‘m outta love
11	3	Cowboys & Kisses
12	1	Shine on you crazy diamond
13	1	Paid my dues

CODD'S NORMAL FORMS FOR DB RELATIONS: 3NF

Third Normal Form (3FN):

- In 2nd normal form
- No functional dependencies between non key fields: a **non-key attribute is dependent from a PK** only

CODD‘S NORMAL FORMS FOR DB RELATIONS: 3NF

CD_ID	Album	Performer	Founded
11	Not that kind	Anastacia	1999
12	Wish you were here	Pink Floyd	1964
13	Freak of Nature	Anastacia	1999

CD_ID	Track	Titels
11	1	Not that kind
11	2	I‘m outta love
11	3	Cowboys & Kisses
12	1	Shine on you crazy diamond
13	1	Paid my dues

CD_ID	Album	Performer
11	Not that kind	Anastacia
12	Wish you were here	Pink Floyd
13	Freak of Nature	Anastacia

Performer	Founded
Anastacia	1999
Pink Floyd	1964

CODD'S NORMAL FORMS - SUMMARY FROM 1NF TO 3NF

CD_ID	Album	Founded	Titels
11	Anastacia – Not that kind	1999	1. Not that kind, 2. I'm outta love, 3 Cowboys & Kisses
12	Pink Floyd – Wish you were here	1964	1. Shine on you crazy diamond
13	Anastacia – Freak of Nature	1999	1. Paid my dues

CD_ID	Album	Performer
11	Not that kind	Anastacia
12	Wish you were here	Pink Floyd
13	Freak of Nature	Anastacia

n

Performer	Founded
Anastacia	1999
Pink Floyd	1964

n

CD_ID	Track	Titels
11	1	Not that kind
11	2	I'm outta love
11	3	Cowboys & Kisses
12	1	Shine on you crazy diamond
13	1	Paid my dues

WHY DATA MODELING?

WHY DATA MODELING?

"Data modeling is the process of learning about the data, and regardless of technology, this process must be performed for a successful application."

Source quote: Steve Hoberman: Data Modeling for Mongo DB, Technics Publications 2014

- Learn about the data and promote collective data understanding
- Derive security classification and measures
- Design for performance
- Accelerate development
- Improve Software quality
- Reduce maintenance costs
- Generate code
- NoSQL Schema-on-read: understand model versions after years

CONCEPTUAL – LOGICAL – PHYSICAL LEVEL

Different levels of abstraction:

- Conceptual (domain) model
  - Focus on (main) entities and its business definitions!
  - No attributes
- Logical design
  - Relational data model (independent of a DBMS or technology)
  - Logic can't affect performance = no performance optimization on this level
- Physical implementation
  - Representation of a data design for a specific DBMS
  - RDBMS are the closest to physical independance

CONCEPTUAL AND LOGICAL LEVEL

Scott Ambler – Disciplined agile delivery

- Do you need it?
- What do you want to achieve?
- What is the value?
- Which representation do you use: 3NF/UML/Object model/ADAPT/Data Vault?

DATA MODELING - WHAT ABOUT DATA MODELING TRAINING?

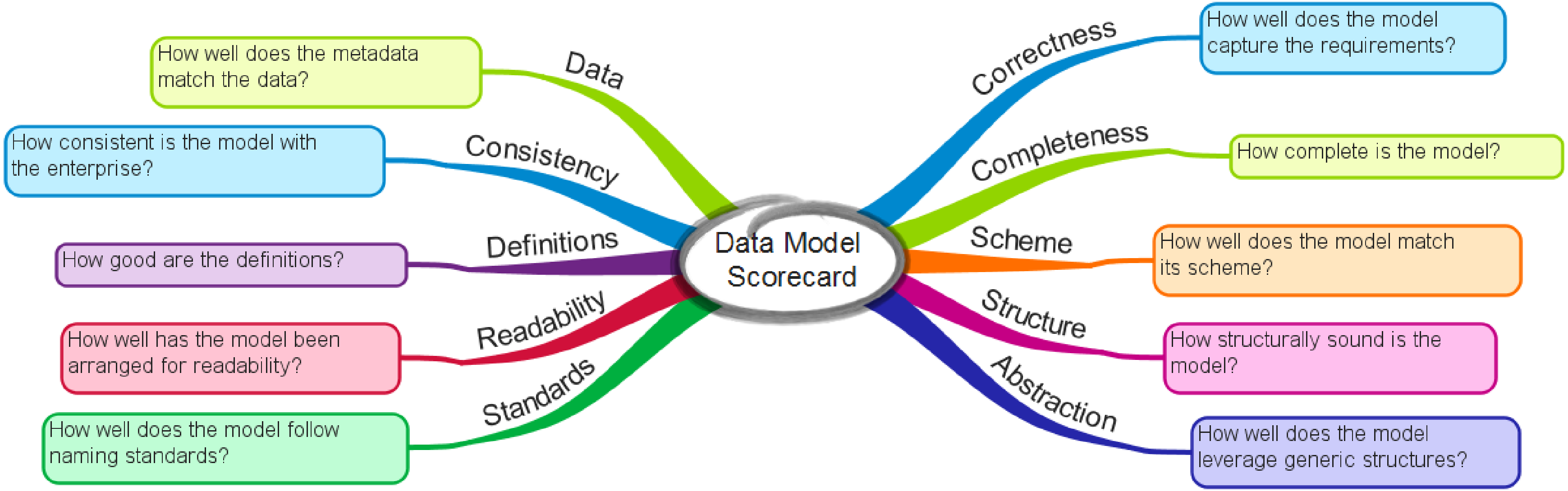
Employees often get trained in SQL Server, Oracle, Cognos TM1, Tableau, or any other tool / product. What about data model training?

To Laugh or Cry?

"I am pretty new in database administration and i am currently working on a project for creating an automating managing system for a school. So I am working in database design. Can someone please tell me how many table i need and if possible the name of all these tables?" --What is the number and the name of all tables to use for a school managing system? --StackExchange.com

Sources: http://www.dbdebunk.com/2017/06/this-week.html

MEASURING THE QUALITY OF A DATA MODEL
DATA MODEL SCORECARD

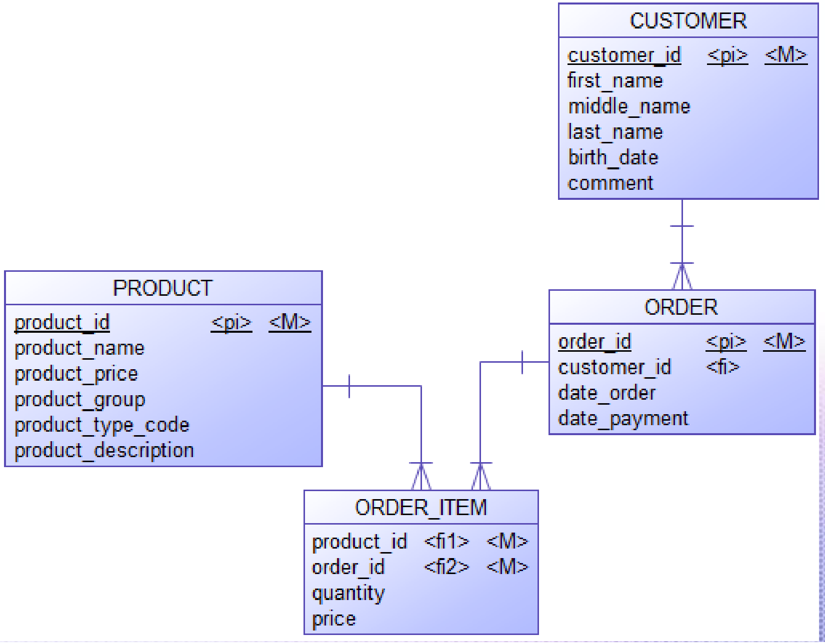


Source: Steve Hoberman - Data Modeling Scorecard, Technics Publication 2015

EXERCISE: OLTP DATA MODEL FOR DWH

The diagram shows a typical OLTP data model

- Customers and products have unique ids and some descriptive attributes
- A customer can place an order on a specific date
- The order contains one or more products

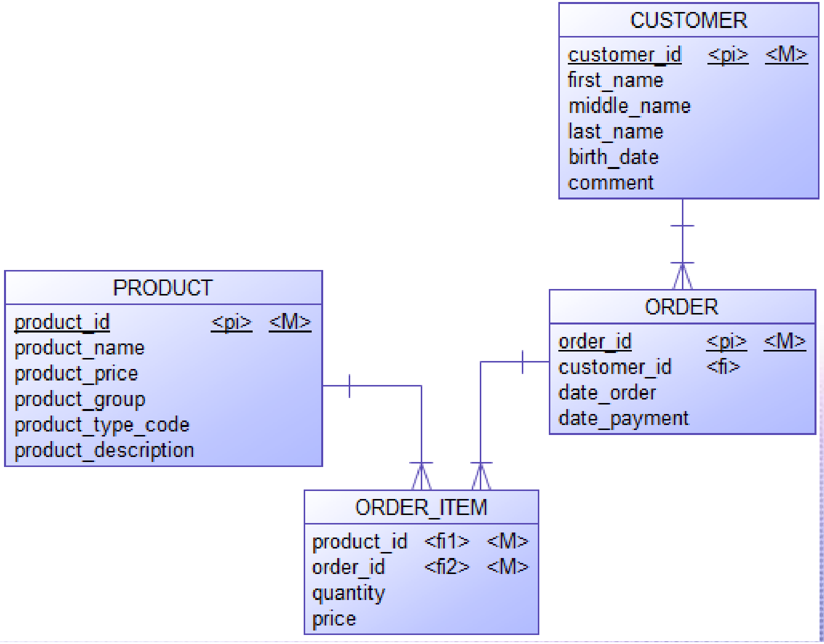


EXERCISE: OLTP DATA MODEL FOR DWH

Now consider DWH requirements like non-volatile and time-variant data

- Customer Bush marries and takes her husband's last name
- Product number 5 gets a price increase

How would you solve such requirements in a data model for the Core Warehouse Layer?

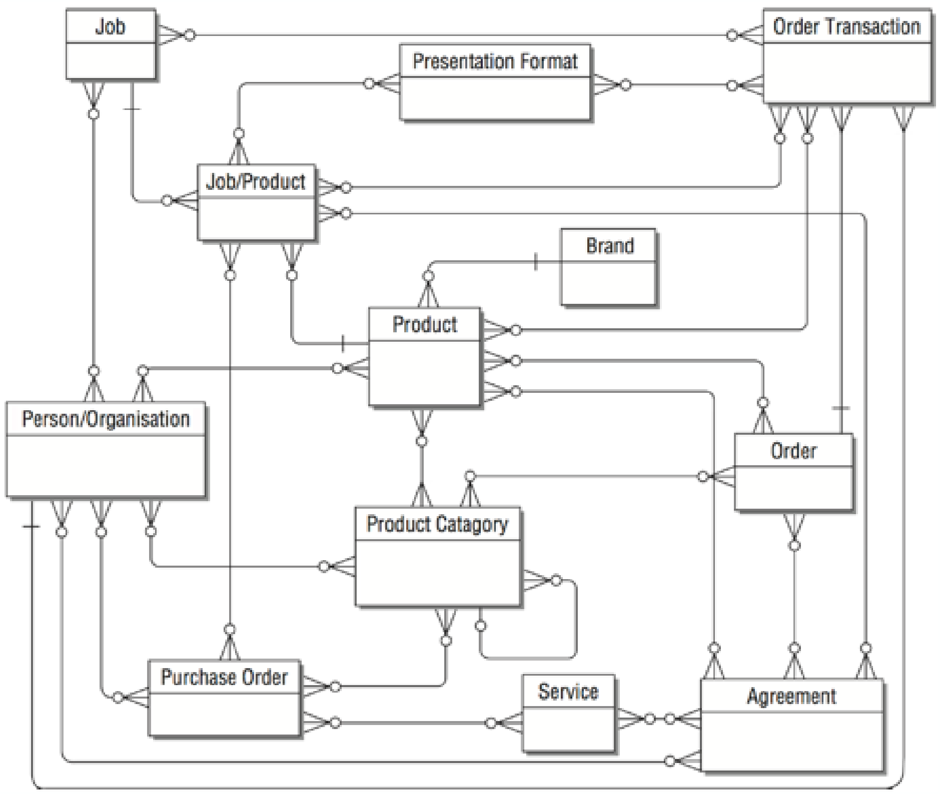


EXERCISE: OLTP DATA MODEL FOR DWH

Possible solutions:

- Add timestamp column as part of the primary key
  - For all tables, not only for specific tables (e.g. product, customer)
  - Composite keys can become inefficient and impractical
- New tables with head and version data to avoid redundancy
  - Head table contains static data that does not change (e.g. customer id, birthdate)
  - Version table contains data that changes (e.g. last name, comments)
- Store every change in log tables
  - Querying tables can become difficult and slow if history is required ("main" table + log tables)

THE CASE AGAINST 3NF FOR DWH



Create a SQL statement for:

How many "Order Transactions" have been created by "Person/Organisation"?

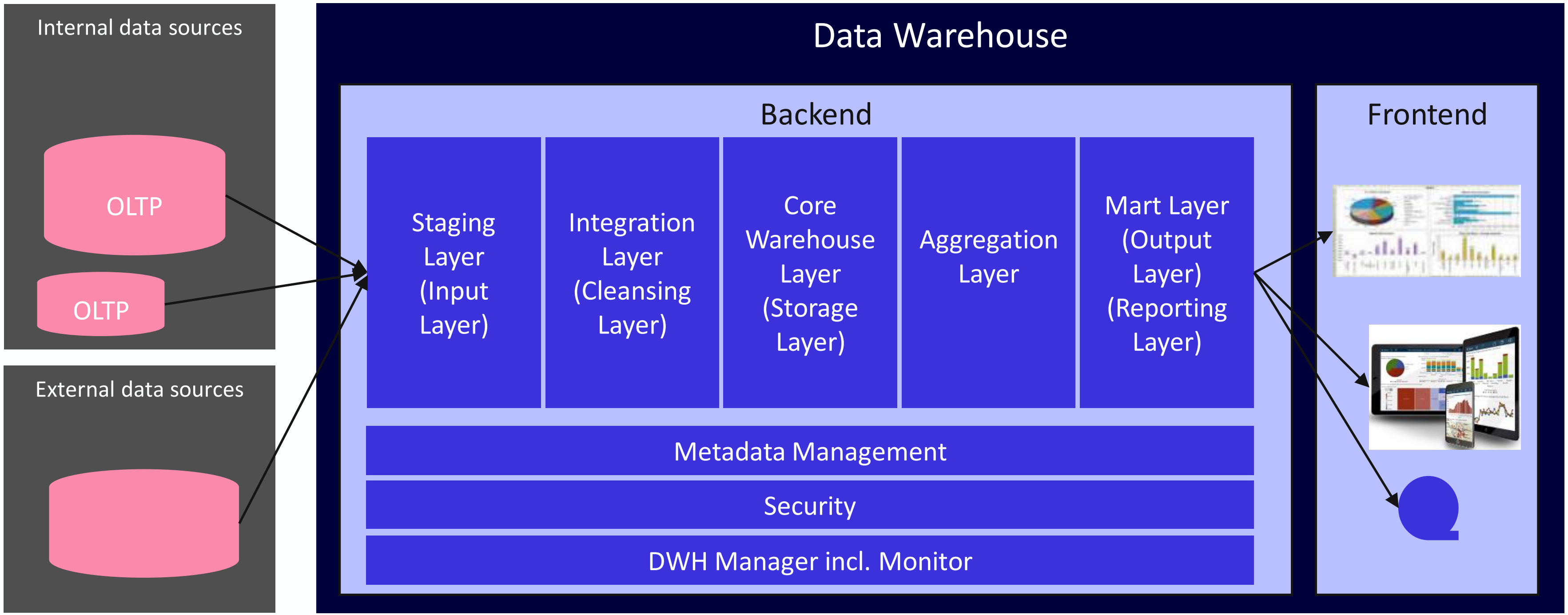
Source: Corr / Stagnitto: Agile Data Warehouse Design, DecisionOne Press, 2011, page 5

DISADVANTAGES OF 3NF FOR DWH

- 3NF is inefficient for query processing
- 3NF models are difficult to understand
- 3NF gets even more complicated with history added
- 3NF not suited for „new“ data sources (JSON, NoSQL, etc.)

→ DWH needs own data modeling approaches for the Core Warehouse Layer and the Mart Layer

LOGICAL STANDARD DATA WAREHOUSE ARCHITECTURE



DATA MODELING IN THE CORE WAREHOUSE LAYER

DATA MODELS IN THE DWH

Layer	Characteristics	Data Model
Staging Layer	▪ Temporary storage ▪ Ingest of source data	▪ Normally 1:1 copy of source table structure – usually without constraints and indexes
Core Warehouse Layer	▪ Historization / bitemporal data ▪ Integration ▪ Tool-independent ▪ Non-redundant data storage ▪ Historization	▪ 3NF with historization ▪ Head and Version modelling ▪ Data Vault ▪ Anchor modeling ▪ Dimensional model with historization (possible)
Data Mart Layer	▪ Performance for end user queries required, Tool-dependent ▪ Lots of joins necessary to answer complex questions	▪ Flat structures, esp. Dimensional model (ROLAP / MOLAP / HOLAP)

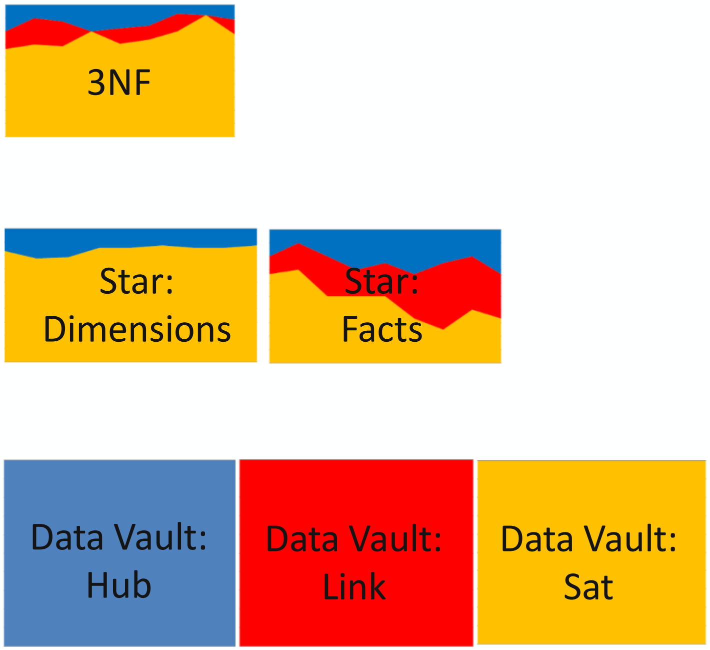
DATA MODELING: 3NF, STAR SCHEMA, DATA VAULT

Vehicle data
Vehicle identifier
Manufacturer
Model
Type
Plant
Delivery Date
Production Date
Price
Source system
Load date/time
Buyer
Salesperson
Engine

Business Key

Context and history

Relationships



DATA VAULT - ARCHITECTURE, METHODOLOGY, MODEL

Architecture

- Multi-Tier
- Scalable
- Supports NoSQL

Lecture part 1:
DWH Architectures

Methodology

- Repeatable
- Measureable
- Agile

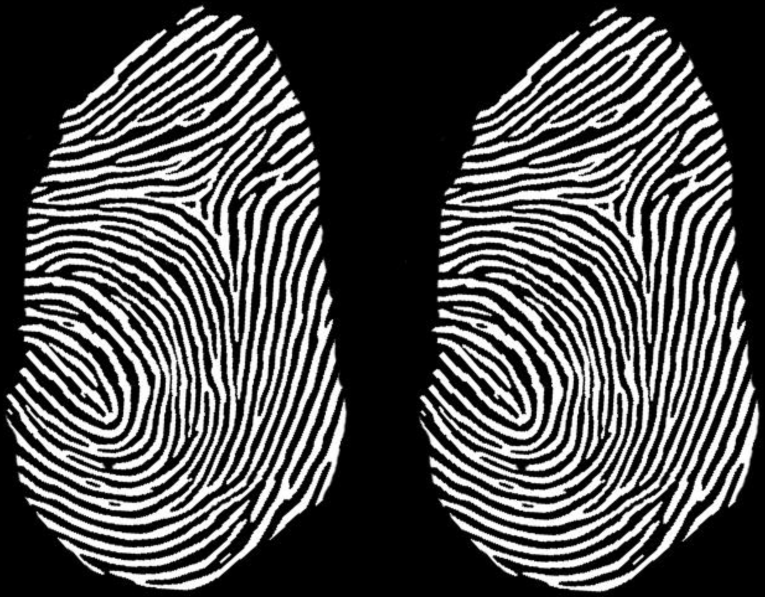
Implementation: Automation, Pattern based, High speed

Model

- Flexible
- Hash based
- Hub & Spoke

Lecture part 2:
DWH Data Modeling

HUB



Unique

identification

by

Natural keys

(Business Keys)

STRUCTURE HUB TABLES

HUB_FAHRZEUG		
VEHICLE_HK	CHAR(32)	<pk>
FIN	VARCHAR(17)	
LOADDATE	TIMESTAMP	
RECORDSOURCE	INTEGER	

HUB_MOTOR		
MOTOR_HK	CHAR(32)	<pk>
MOTORNUMMER	VARCHAR(50)	
LOADDATE	TIMESTAMP	
RECORDSOURCE	INTEGER	

HUB_STEUERGERAET		
STEUERGERAET_HK	CHAR(32)	<pk>
SERIENUMMER	VARCHAR(50)	
LOADDATE	TIMESTAMP	
RECODSOURCE	INTEGER	

HUB TABLES: TYPICAL CHARACTERISTICS

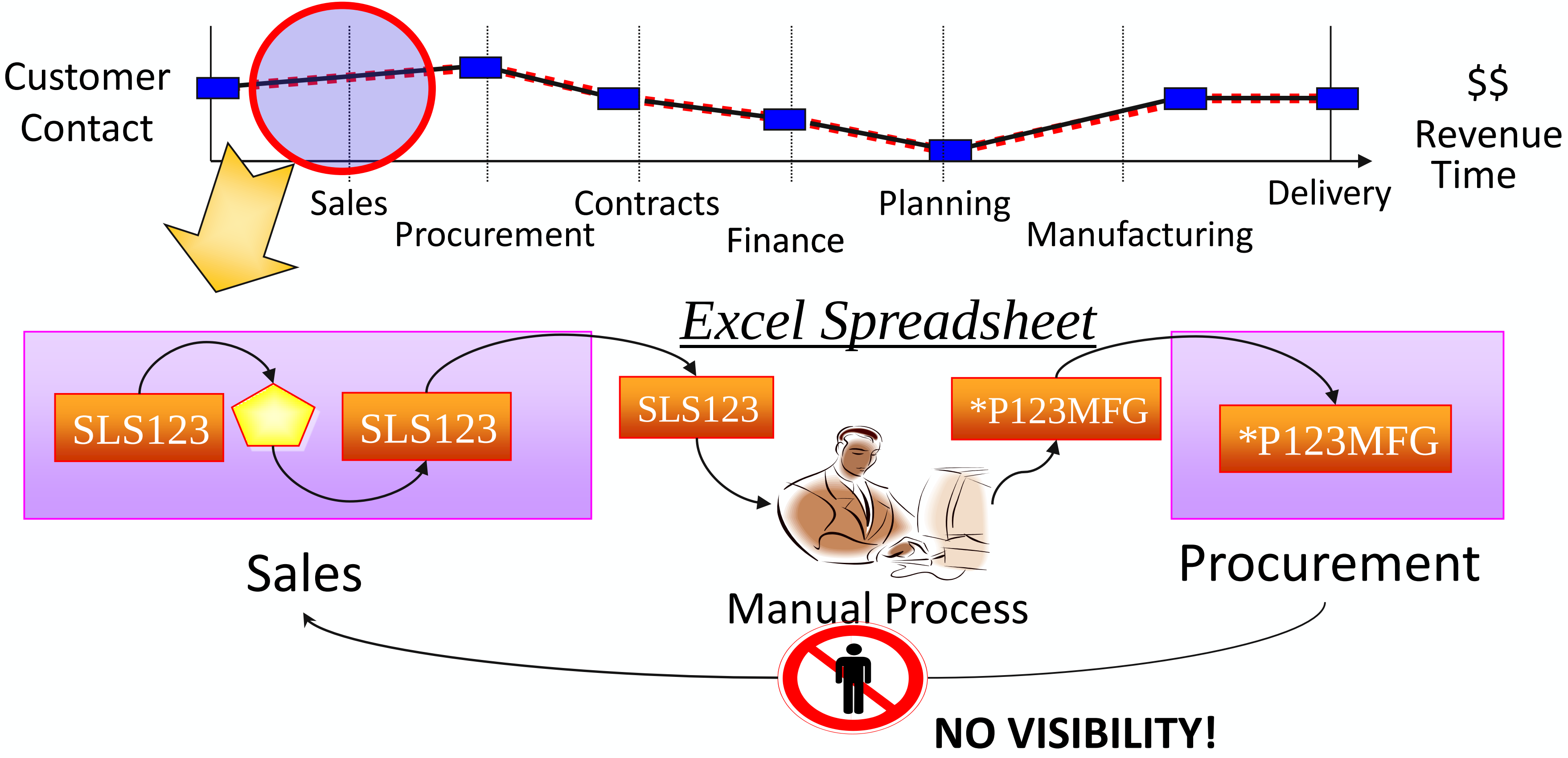
Business Keys should be natural keys used by the business (e.g. Vehicle Identifier, Serial number)

Business Keys should stand alone and have meaning to the business

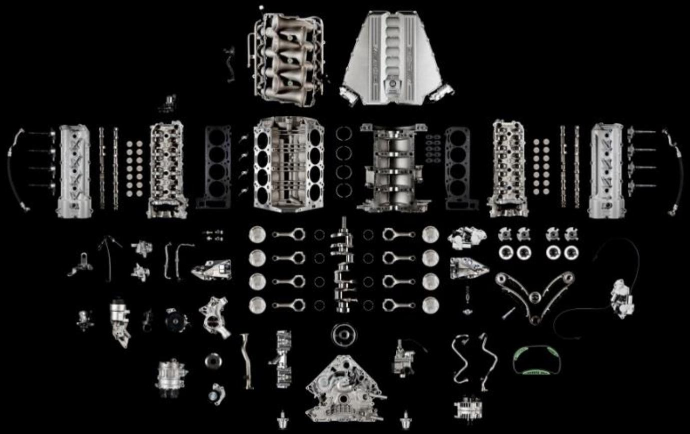
Business Keys should **never** change, have the same semantic meaning and the same granularity

Focus on Business Keys (instead focus on source system surrogates) ensures that the result serves the needs of the business

TYING BUSINESS PROCESSES TO BUSINESS KEYS

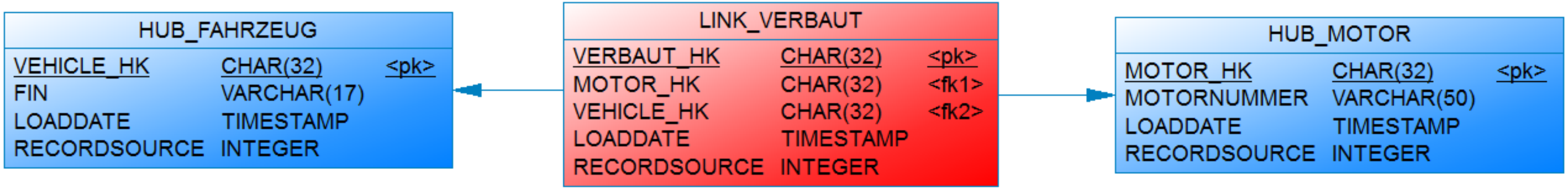


LINK



Unique relationships between Business Keys (HUBs)

STRUCTURE LINK TABLES



LINK TABLES: TYPICAL CHARACTERISTICS

A LINK models a relationship between 2 or more HUBs

The relationship is always n:m

The composed key must be unique. One of the foreign keys is driving key

Link to Link allowed but should be avoided in a physical implementation due to load dependency

CANDIDATES FOR LINKS

- Relationships / Associations
  - Foreign Keys in OLTP systems
- Hierarchies and Redefinitions
  - Hierarchical relationships are modeled by one link and two connections to HUBs: HAL (parent-child LINK) and SAL (same-as LINK)
- Transactions and events are often modeled as link (could also be a Hub)
  - E.g. sales order or sensor data
  - Intensive discussions about modeling as Hub or Link on conferences or social media (modeling solution depends from requirements, context, etc)

SAT

Descriptive,

detailled,

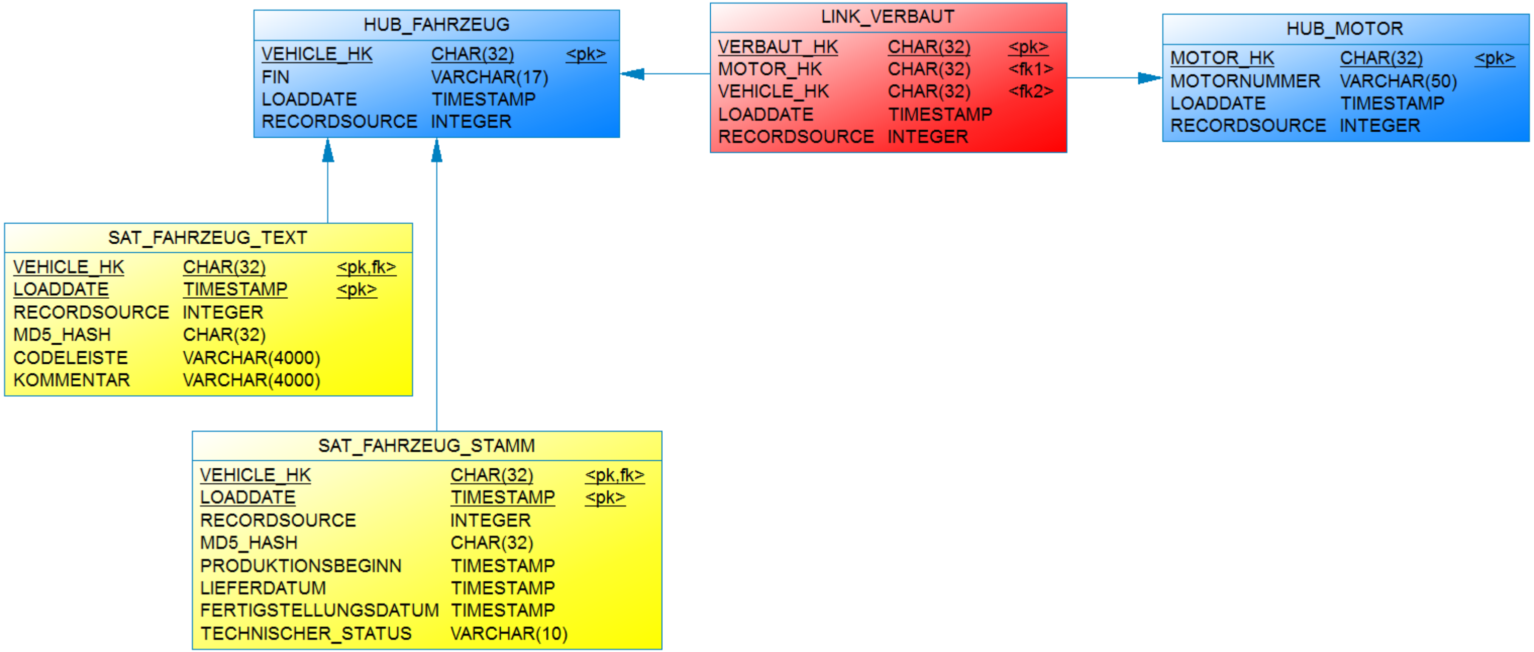
current

and

historized

data

STRUCTURE SAT TABLES



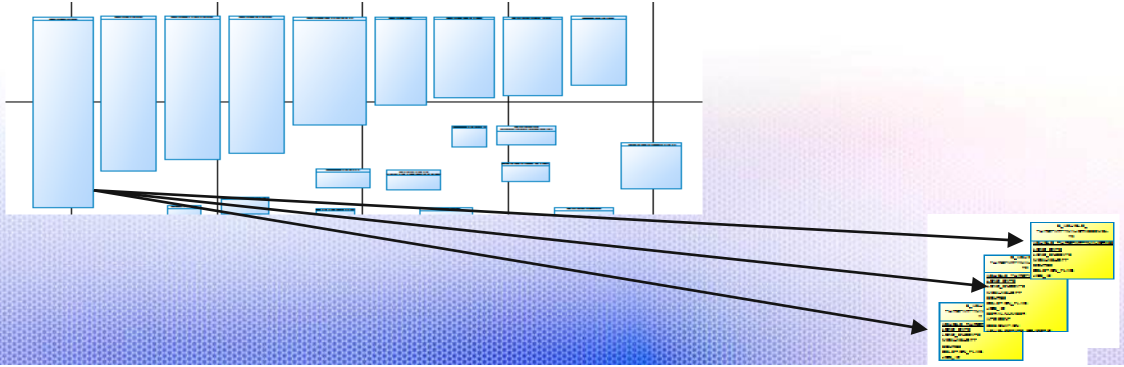
SAT TABLES: TYPICAL CHARACTERISTICS

- Contains all non-key attributes
- Is connected to exactly one Hub or Link
- HUB or LINK tables can (should) have several SAT tables, e.g. by source system
- Can contain in the extreme case one column only (or any number of columns)

SAT TABLE DESIGN

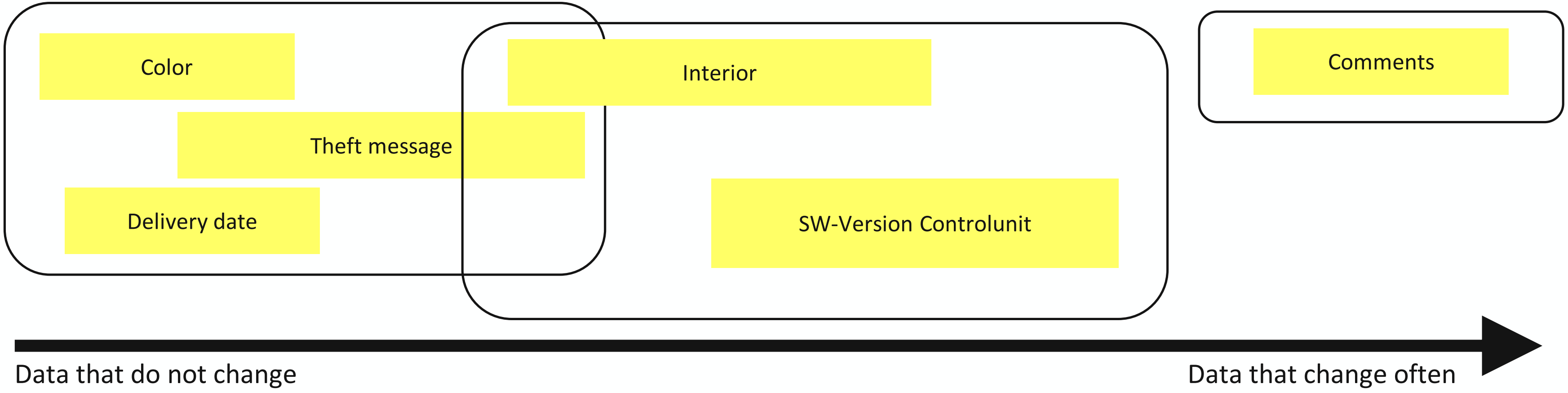
Different criteria to design SAT tables (separate data into different SAT tables)

- Source system
- Rate of change
- Data types (e.g. separate CLOBS or other lengthy textual fields)



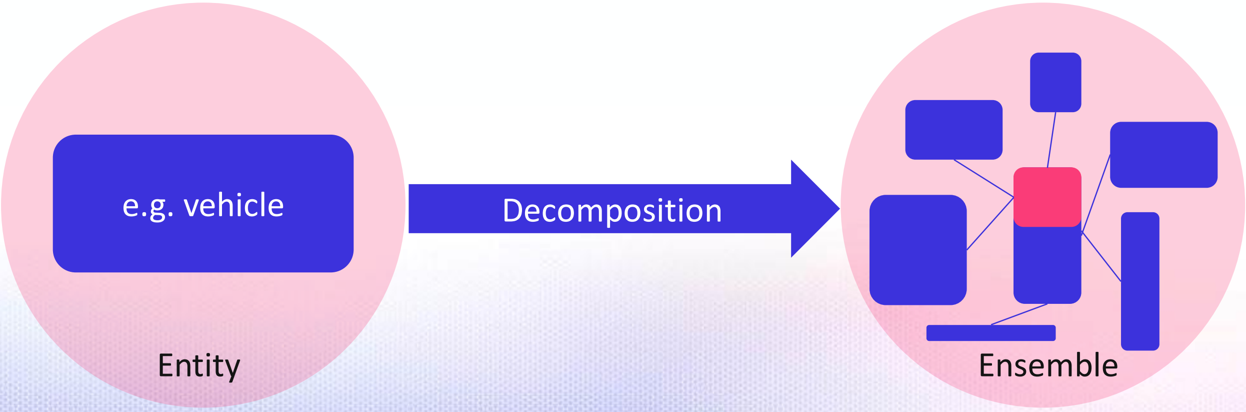
SAT TABLE DESIGN

Rate of change in order to avoid redundant storage of data

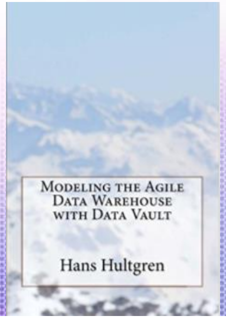


ENSEMBLE MODELING

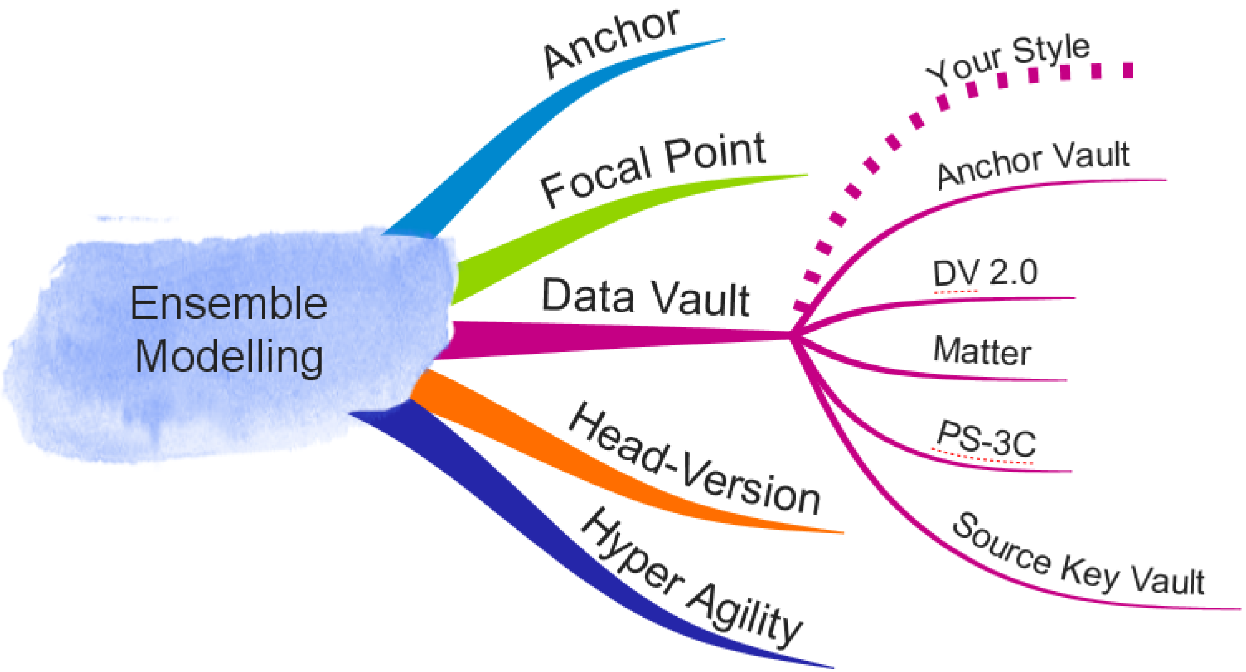
Hans Hultgren: “An ensemble is a representation of a Core Business Concept including all of its parts – the business key, with context and relationships”



Source:



ENSEMBLE MODELING – NOT JUST DATA VAULT 2.0

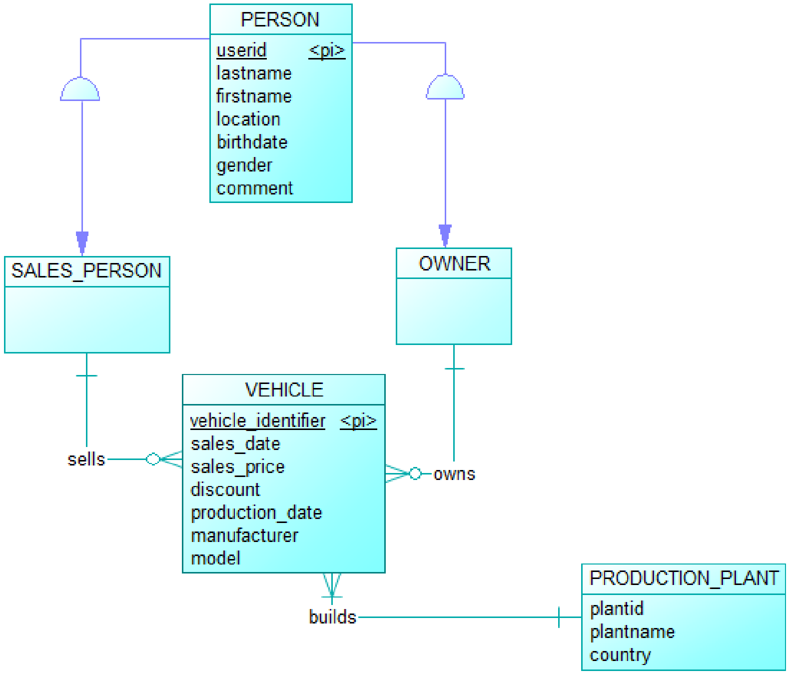


EXERCISE DATA VAULT

The following data model shows vehicle sales with entities

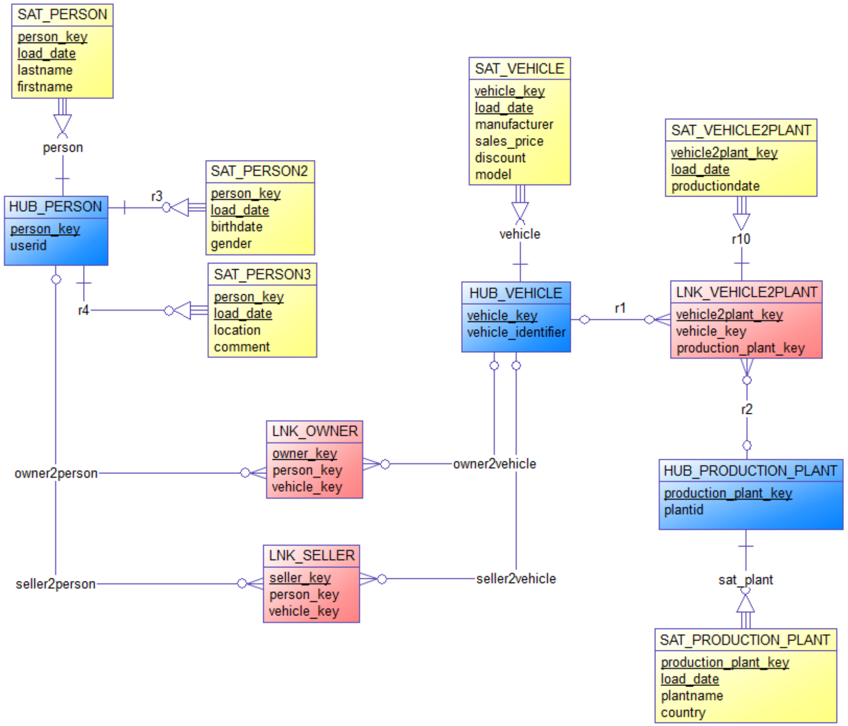
- Person (sales_person and owner)
- Vehicle
- Production_plant

Architect a Data Vault model for the Core Warehouse Layer



Report 1: sum(sales_price) by sales_person and vehicle_type
Report 2: count(vehicle) by plantname
Report 3: sales by sales_person

SAMPLE SOLUTION DATA VAULT



DATA VAULT - ADVANTAGES

- Flexible / agile approach
- Highly parallel data loads, Scalable
- Automatable
- Systematic approach that covers historization and integration
- Full auditability
- No updates or deletes on business data
- Horizontal and vertical partitioning
- Supports / Combines RDBMS and Hadoop/NoSQL technologies

DATA VAULT - DISADVANTAGES

- More Tables
- More joins
  - Performance to load Data Mart can be a challenge
  - Logic to load Data Marts can be rather complex if many Data Vault tables are involved
- All relationships are modeled n:m (documentation necessary!)
- The same source table is used several times while loading HUBs, SATs, LINKs

DIMENSIONAL DATA MODELING IN THE MART LAYER: ROLAP AND MOLAP

DATA MODELS IN THE DWH

Layer	Characteristics	Data Model
Staging Layer	▪ Temporary storage ▪ Ingest of source data	▪ Normally 1:1 copy of source table structure – usually without constraints and indexes
Core Warehouse Layer	▪ Historization / bitemporal data ▪ Integration ▪ Tool-independent ▪ Non-redundant data storage ▪ Historization	▪ 3NF with historization ▪ Head and Version modelling ▪ Data Vault ▪ Anchor modeling ▪ Dimensional model with historization (possible)
Data Mart Layer	▪ Performance for end user queries required, Tool-dependent ▪ Lots of joins necessary to answer complex questions	▪ Flat structures, esp. Dimensional model (ROLAP / MOLAP / HOLAP)

DIMENSIONAL MODELING

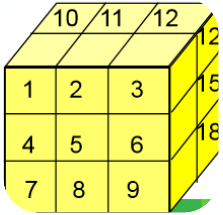
- Design technique to present data in a standard, intuitive framework
  - **Easily understandable for end users**
  - **High performance end user access**
  - Logical data model
  - Physical data model: Not necessarily relational, can also be stored in specialiced multi-dimensional tools ("OLAP Cubes")
- Analysis / Reporting of numerical **measures** (metrics) by different **attributes (context)**

DIMENSIONAL MODEL – IMPLEMENTATION TYPES

Implementation types of dimensional models

Star Schema = Relational model (ROLAP) consists of

- Fact Tables
- Dimension Tables



Cube = Multidimensional model (MOLAP) consists of

- Edges = Attributes
- Cells = Measures (facts)

DIMENSIONAL MODEL

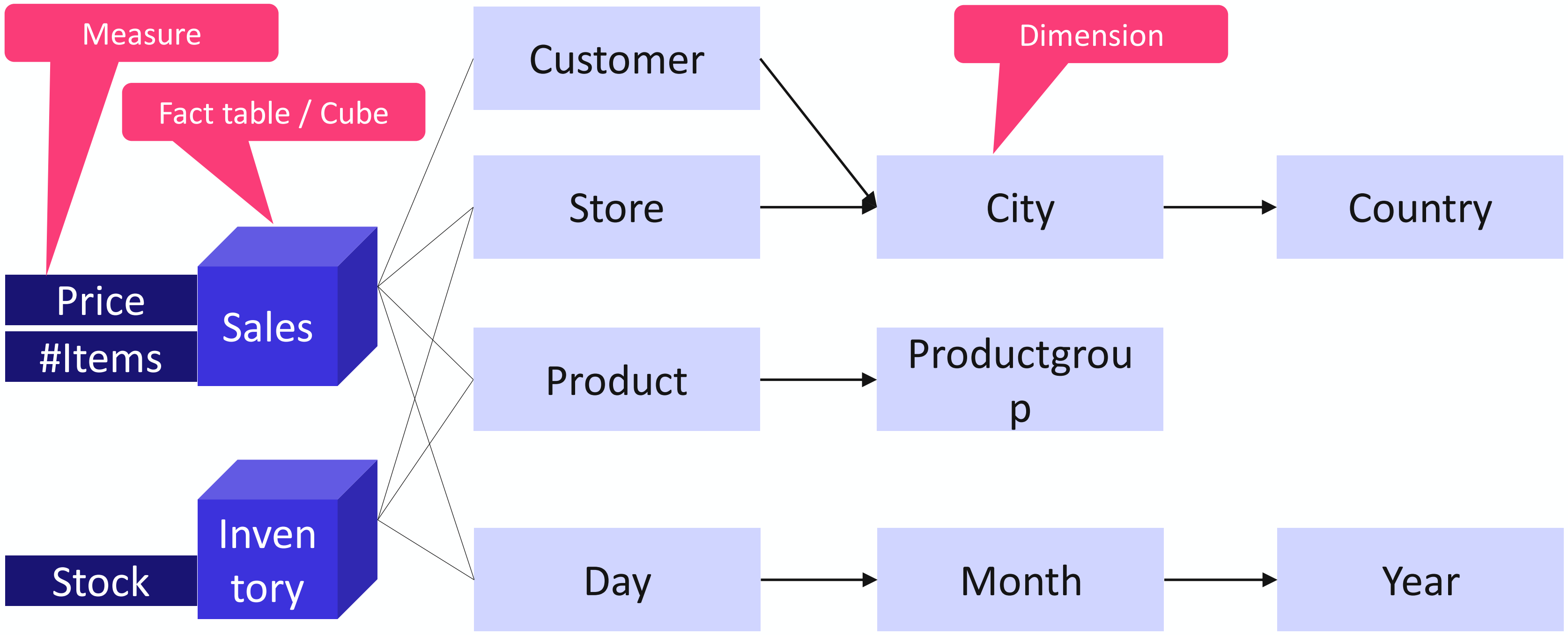
Dimensions

- Are entities that contain descriptive textual attributes for analysis
  - E.g. Car (model, manufacturer, etc), Time period (day, week, month, year)

Facts

- Contain key numerical figures – “Measures” – “Metrics”
  - E.g. Sales amount
    (for dimensions: product X in region y and time period z)

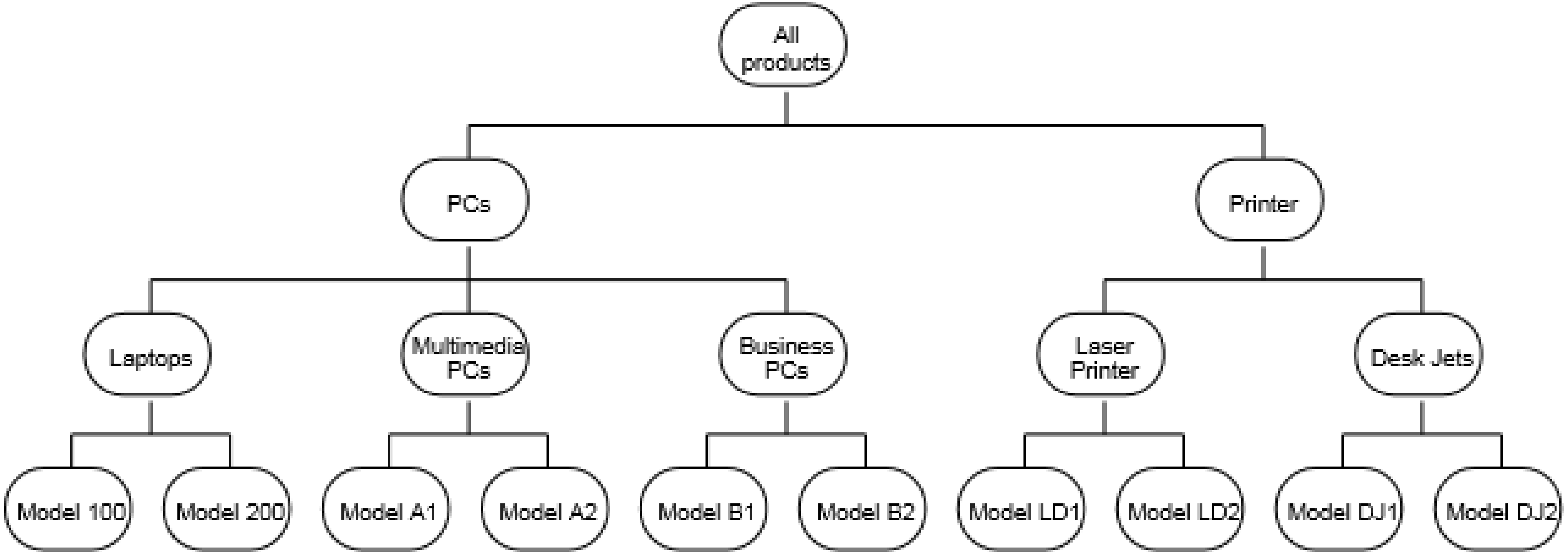
DIMENSIONAL MODEL – LOGICAL VIEW



SAMPLE PRODUCT HIERARCHY

Dimensions can be organized in hierarchies

- i.e. product hierarchy



HIERARCHIES

Other hierarchies:

- Date → Month/Year → Quarter/Year → Year
- Customer → Company → Industry
- City → County/Landkreis → State → Country → Continent

Arbitrary number of hierarchy levels

Purpose:

- group and structure data
- enable view on data at different levels of granularity
- Hierarchies define aggregations on measures

ROLAP

Implementation types of dimensional models

Star Schema = Relational model (ROLAP) consists of

- Fact Tables
- Dimension Tables

Cube = Multidimensional model (MOLAP) consists of

- Edges = Attributes
- Cells = Measures (facts)

ROLAP

Physical data structure: relational tables

- Advantage: can use well-engineered, reliable and high-performance database systems and query languages

Special table structure

- Star / Snowflake Schema
  - Dimension tables with textual attributes
  - Fact table with measures consisting of foreign keys to dimension tables

ROLAP

Special table structure (continued)

- Memory amount **depends mainly on the number of facts**
  - One row per fact
  - Size of a row approx. (#dimensions + #measures) * column size
- Aggregated totals are **computed dynamically** in general
  - Longer response times

RELATIONAL DATA MODEL

Dimensions

- Relational table for each dimension like product, region, time period
- Primary key (**surrogates**) identifies each dimension element
- Additional **fields contain descriptive information** like product name
  - E.g. Dimensions: Product, Region, Time period (day, week, month, year)

Facts

- Relational table containing key figures – “Measures”
- Stores **foreign keys** to dimension tables
- The other fields contain the **values of the key figures/measures**
  - E.g. Sales amount (for product X in region y and time period z)

RELATIONAL MODEL: STAR SCHEMA

DATA MODELS FOR HIERARCHIES

Denormalized Dimensions

- 1 Table with all hierarchy levels
- Advantage:
  - Efficient aggregations
  - Performance
- Disadvantage:
  - Complex updates if hierarchies change

Productid	Productname	Productgroup	Productcategory	Productclass
1234ABC	Thinkpad T60	Laptop	PC	Computer

DATA MODELS FOR HIERARCHIES

Normalized Dimensions

- 1 table for each hierarchy level
- Advantage:
  - Minimal updates for changes in the hierarchies
- Disadvantage:
  - More complex queries when computing aggregations
  - Multiple joins

Productid	Productname	Productgid
1234ABC	ThinkPad T60	G1234

↓

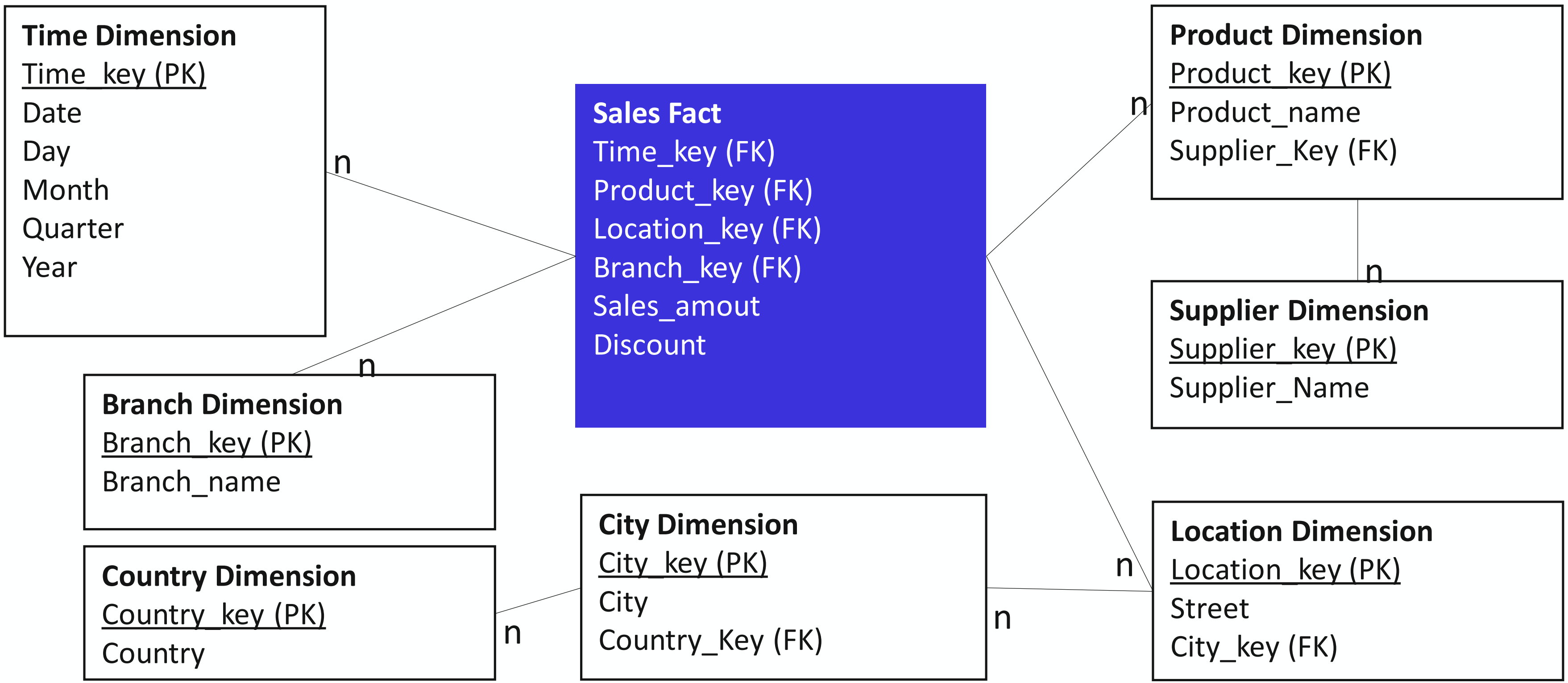
Productgid	Productgroup	Productcatid
G1234	Laptop	CAT12

↓

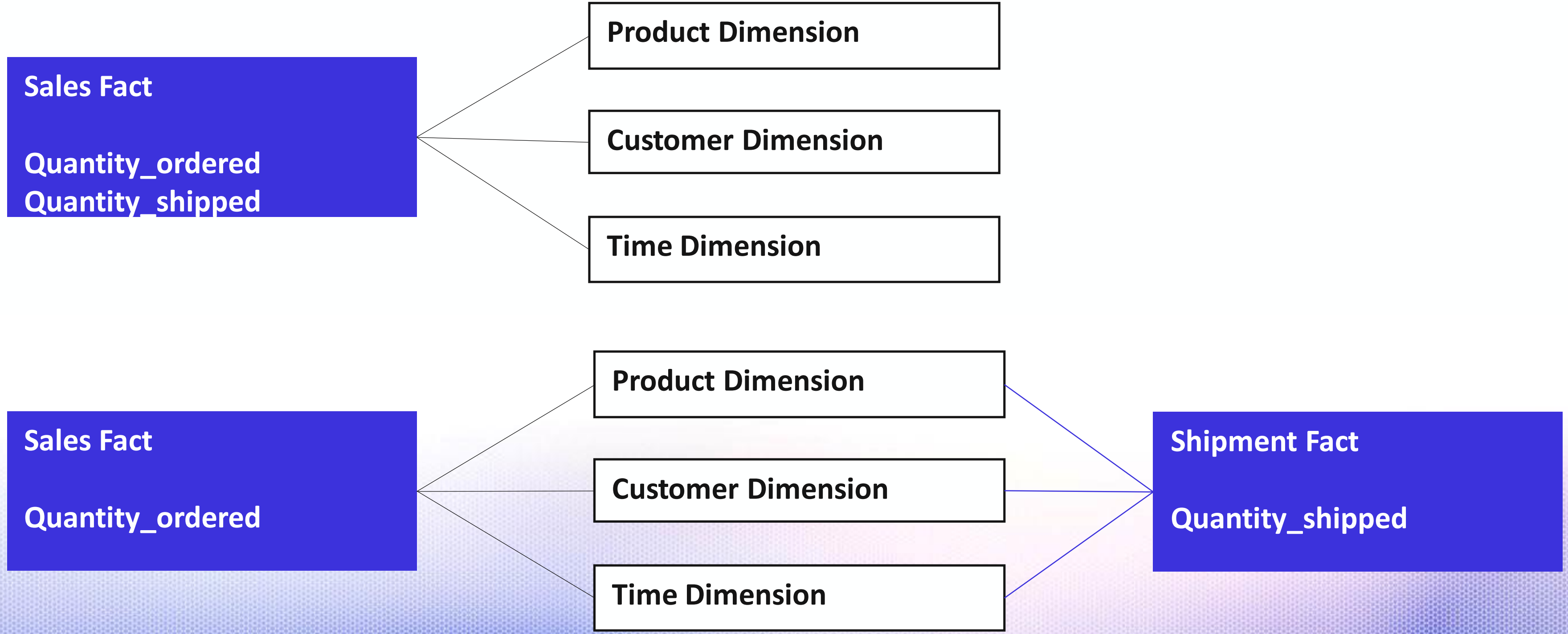
↓

Productclassid	Productclass
C3	Computer

RELATIONAL MODEL: SNOWFLAKE SCHEMA WITH NORMALIZED DIMENSIONS



ONE OR TWO FACT TABLES?



ONE OR TWO FACT TABLES?

Time	Product	Customer	Quantity ordered	Quantity shipped
1	A	X	100	NULL
1	B	Y	50	NULL
2	A	X	NULL	100

- Reports get much more complicated to filter NULL
    - Avg(quantity): 100+50/2 but avg(shipped): 100/1
- There may be even more columns like quantity_delivered or Delivery_company
- → 2 fact tables

ONE OR TWO FACT TABLES?

Different processes must result into different fact tables

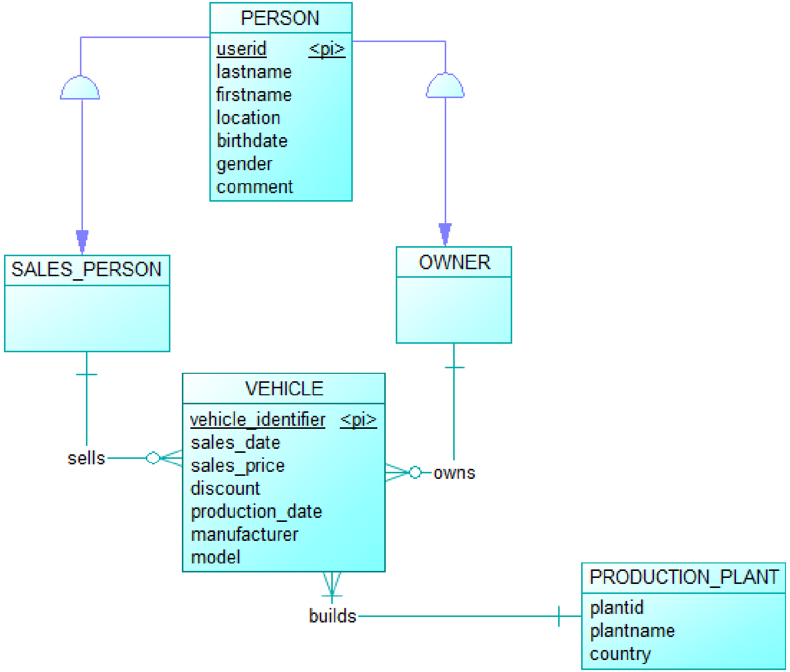
- E.g. measures at different time
- E.g. facts with different grain

EXERCISE STAR SCHEMA

The following data model shows vehicle sales with entities

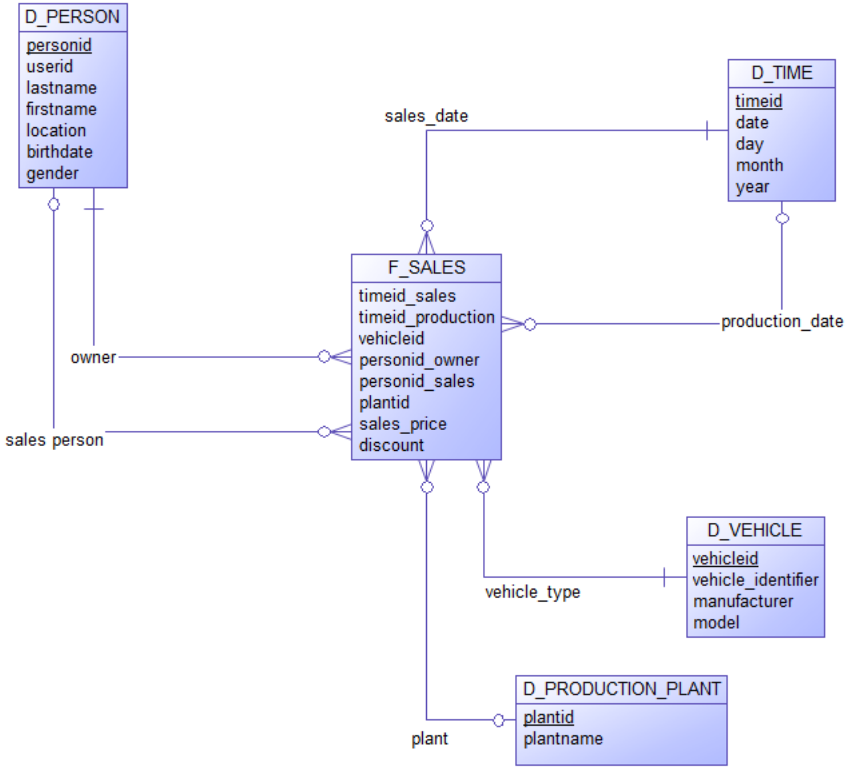
- Person (sales_person and owner)
- Vehicle
- Production_plant

Architect a Star Schema for the Data Mart Layer



Report 1: sum(sales_price) by sales_person and vehicle_type
Report 2: count(vehicle) by plantname
Report 3: sales by sales_person

SAMPLE SOLUTION STAR SCHEMA



ROLAP ENHANCEMENTS

Used for accelerating data warehouse queries in general

- **Precomputation** of aggregated values
  - Materialized views / query tables store data physically
- **Relational Columnar (in-memory) databases**

PRECOMPUTATION OF AGGREGATED TOTALS

Query processing in the Mart Layer

- SQL statements **can become complex**, e.g. many joins
- SQL statements **can become slow** if many rows are aggregated
  - E.g. sum of sales amount for city X AND product Y AND year 2016 compared to city X AND product Y AND year 2015
- If aggregated values are stored in Fact tables, new data from the Core Warehouse layer have to be integrated into such aggregated fact tables

MATERIALIZED VIEWS/QUERY TABLES

The DBMS takes care of solving these problems

- The user defines views containing aggregated values for certain hierarchy levels
- These views are **materialized as tables**
  - Update options
    - immediate
    - deferred
- When performing a query against a fact table the DB optimizer takes advantage of these materialized views, i.e., no special queries have to be written for this by a user or application program
  - The user has **not** to rewrite the original query to use the materialized views

MATERIALIZED VIEWS / MATERIALIZED QUERY TABLES

Example statement Oracle to precompute values (similar DB2 and other RDBMS)

```
CREATE MATERIALIZED VIEW sales_agg
BUILD IMMEDIATE
REFRESH FAST
ON DEMAND
AS
SELECT p.productname, s.city, EXTRACT(MONTH FROM s.date)
     , sum(s.sales_amount)
     , sum(no_items)
FROM    product p
JOIN    sales s ON p.productid = s.productid
GROUP by p.productname, s.city, EXTRACT(MONTH FROM s.date);
```

RELATIONAL COLUMNAR DATABASES

Row-oriented storage

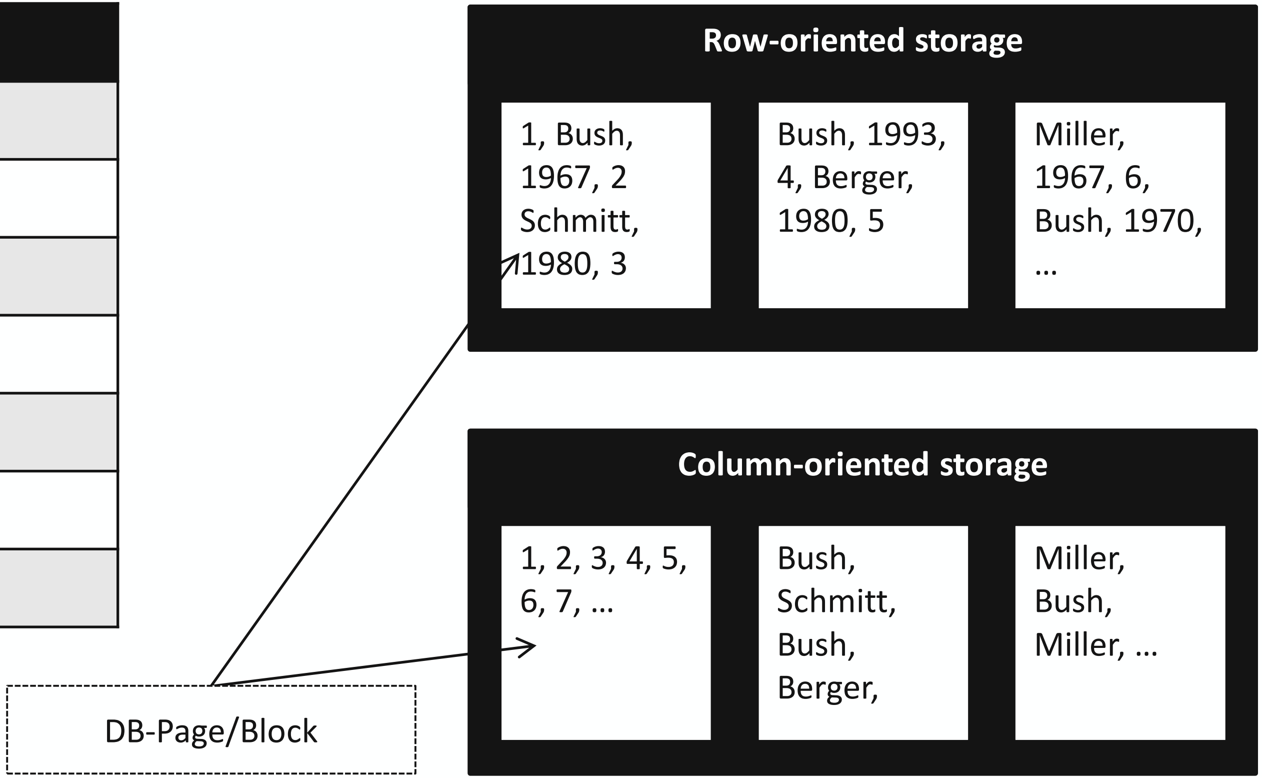
- Data of a relational table is stored row wise:
  <values of Row 1><values of Row 2> … <values of Row N>

Column-oriented storage

- The values of each column are stored separately:
  <values of Column 1><values of Column 2> … <values of Column M>

ROW AND COLUMN ORIENTED DB BLOCK STORAGE

Id	Name	Birthdate
1	Bush	1967
2	Schmitt	1980
3	Bush	1993
4	Berger	1980
5	Miller	1967
6	Bush	1970
7	Miller	1980



ROW VS COLUMN ORIENTED STORAGE

Row-oriented storage

- Data of one row is grouped on disk and can be retrieved through one read operation
- Single values can be retrieved through efficient index and off-set computations
- Good Insert, update and delete operations performance
- → **Suited for OLTP systems**

ROW VS COLUMN ORIENTED STORAGE

Column-oriented storage

- Data-of one column is grouped on disk and can be retrieved with far less read operations than for row-oriented storage
- This makes computation of aggregations much faster in particular for tables with a lot of columns
- In general better suited for queries involving partial table scans
- Bad Insert, update and delete operations performance
- Normally excellent compression as identical data types are stored in same blocks
- Products: SAP HANA, HP Vertica, Exasol, IBM DB2 BLU, Oracle In-Memory Option, SQL Server (Columnar Indexes), etc
- → **Suited for OLAP systems**

HOW TO COVER DATA CHANGES IN THE MART?

Data changes, e.g.

- new employees
- employees change departments
- employees leave
- whole department reorganisations, etc

How are the changes handled? Insert-only approach in the Core Warehouse Layer, but choices in the Mart Layer (reduce data amount to what end user needs)

- **What does the business want to see? (Reporting Scenarios)**
- **How is data inserted / updated in dimensions? (Slowly Changing Dimensions)**

REPORTING SCENARIOS

- As-is scenario
- As-of scenario
- As-posted scenario
- As-posted with comparable data scenario

DATA MART – EXAMPLE BASELINE

Employee Dimension 2015

Employee	Organisation
Miller	DWH
Rogers	DWH
Douglas	Database
Powell	Database

Employee Dimension 2016

Employee	Organisation
Miller	DWH
Rogers	DWH
Powell	DWH
Douglas	Database
Bush	Database

Other department

New employee

Facts

Employee	Year	#Pro-jects
Miller	2015	10
Rogers	2015	10
Douglas	2015	10
Powell	2015	10
Miller	2016	10
Rogers	2016	10
Powell	2016	10
Douglas	2016	10
Bush	2016	10

Assumption: current year: 2016

AS-IS SCENARIO

Reporting uses current structure

Employee Dimension 2016

Employee	Organisation
Miller	DWH
Rogers	DWH
Powell	DWH
Douglas	Database
Bush	Database

Facts

Employee	Year	#Pro-jects
Miller	2015	10
Rogers	2015	10
Douglas	2015	10
Powell	2015	10
Miller	2016	10
Rogers	2016	10
Powell	2016	10
Douglas	2016	10
Bush	2016	10

Organisation	#Projects´15	#Projects´16
DWH	30	30
Database	10	20

AS-OF SCENARIO

Reporting uses structure as demanded

e.g. requested for 2015

Employee Dimension 2015

Employee	Organisation
Miller	DWH
Rogers	DWH
Douglas	Database
Powell	Database

Facts

Employee	Year	#Pro-jects
Miller	2015	10
Rogers	2015	10
Douglas	2015	10
Powell	2015	10
Miller	2016	10
Rogers	2016	10
Powell	2016	10
Douglas	2016	10
Bush	2016	10

Organisation	#Projects´15	#Projects´16
DWH	20	20
Database	20	20

AS-POSTED SCENARIO

Reporting uses „historical truth“

Employee Dimension 2015

Employee	Organisation
Miller	DWH
Rogers	DWH
Douglas	Database
Powell	Database

Employee Dimension 201

ee	Organisation
Miller	DWH
Rogers	DWH
Powell	DWH
Douglas	Database
Bush	Database

Facts

Employee	Year	#Pro-jects
Miller	2015	10
Rogers	2015	10
Douglas	2015	10
Powell	2015	10
Miller	2016	10
Rogers	2016	10
Powell	2016	10
Douglas	2016	10
Bush	2016	10

Organisation	#Projects´15	#Projects´16
DWH	20	30
Database	20	20

AS-POSTED WITH COMPARABLE DATA SCENARIO

Reporting uses „historical truth“ for identical dimension data

Employee Dimension 2015

Employee	Organisation
Miller	DWH
Rogers	DWH
Douglas	Database
~~Powell~~	~~Database~~

Employee Dimension

Employee	Organisation
Miller	DWH
Rogers	DWH
~~Powell~~	~~DWH~~
Douglas	Database
~~Bush~~	~~Database~~

Organisation	#Projects´15	#Projects´16
DWH	20	20
Database	10	10

Facts

Employee	Year	#Pro-jects
Miller	2015	10
Rogers	2015	10
Douglas	2015	10
Powell	2015	10
Miller	2016	10
Rogers	2016	10
Powell	2016	10
Douglas	2016	10
Bush	2016	10

SLOWLY CHANGING DIMENSIONS

Dimensions must absorb changes

Slowly changing dimensions according to Kimball / Ross (2002):

- SCD Type 0
  - no changes, new data is ignored
- **SCD Type 1 - 3**
  - See next slides
- And some more SCD types
  - Rarely relevant

SLOWLY CHANGING DIMENSIONS – EXAMPLE BASELINE

Employee Dimension

ID	Employee	Organisation
1	Miller	DWH
2	Powell	Database

Changes:

- New data added: Albert, DWH
- Powell marries and has new name Parker

SLOWLY CHANGING DIMENSION TYPE 1

Employee Dimension

ID	Employee	Organisation
1	Miller	DWH
2	Powell	Database

- No History
- Dimension attributes always contain current data

Employee Dimension

ID	Employee	Organisation
1	Miller	DWH
2	Parker	Database
3	Albert	DWH

Changes:

- New data added: Albert, DWH
- Powell marries and has new name Parker

SLOWLY CHANGING DIMENSION TYPE 2

Employee Dimension

ID	Employee	Organisation
1	Miller	DWH
2	Powell	Database

Employee Dimension

ID	Employee	Organisation	Valid From	Valid To
1	Miller	DWH	01.01.2015	NULL
2	Powell	Database	21.12.2014	15.10.2016
3	Albert	DWH	05.03.2014	NULL
2	Parker	Database	15.10.2016	NULL

- Full Historization
- Dimension contains timestamps with NULLs or future date like 31.12.2999

Changes:

- New data added: Albert, DWH
- Powell marries and has new name Parker

SLOWLY CHANGING DIMENSION TYPE 3

Employee Dimension

ID	Employee	Organisation
1	Miller	DWH
2	Powell	Database

- Historization of latest change only
- And storage of current value

Employee Dimension

ID	Employee Name	Previous Name	Organisation	Previous Organisation
1	Miller	NULL	DWH	NULL
2	Parker	Powell	Database	NULL
3	Albert	NULL	DWH	NULL

Changes:

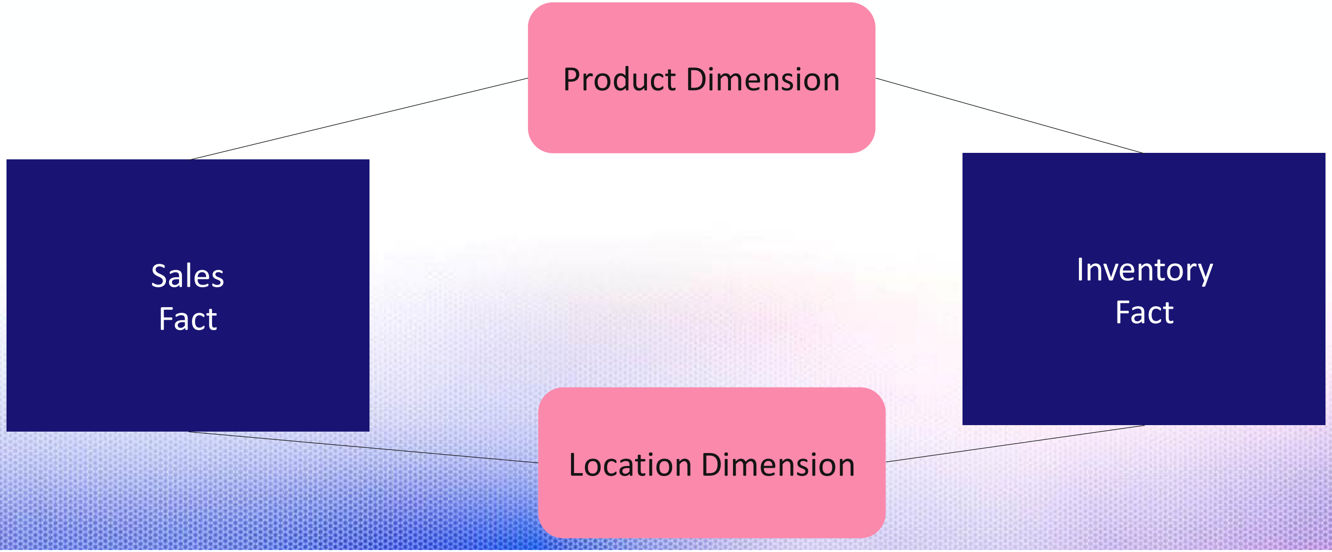
- New data added: Albert, DWH
- Powell marries and has new name Parker

DIMENSION AND FACT TABLE TYPES

- Conformed dimension
- Junk dimension
- Role-Playing dimension
- Degenerated dimension
- Transactional fact
- Periodic fact
- Accumulating fact

DIMENSION TYPES: CONFORMED DIMENSION

- Dimension that is used in several fact tables
- Fact tables can be connected by using conformed dimensions



DIMENSION TYPES: CONFORMED DIMENSION

Kimball: **Enterprise DWH Bus Matrix** is a “design tool” to document the organization’s processes

	Date	Product	Location	Customer	Promotion
Sales Fact	X	X	X	X	X
Inventory Fact	X	X	X		
Customer Returns Fact	X	X	X	X	
Sales Forecast Fact	X	X	X		

DIMENSION TYPES: JUNK DIMENSION

Collection of lookup data / codes that could also form it’s own dimension

ID	MartialStatus	Gender
1	Single	Male
2	Single	Female
3	Married	Male
4	Married	Female

DIMENSION TYPES: ROLE-PLAYING DIMENSION

A single dimension is referenced several times by the same fact table

- E.g. several dates in fact table reference Date Dimension

ID	OrderDate	DeliveryDate	ProductionDate
1	..	..	..
2	..	..	..
3	..	..	..
4	..	..	..

DIMENSION TYPES: DEGENERATED DIMENSION

- A dimension without own dimension table. Data are stored in the fact table only.
- Used e.g. for drill-through in reports
  - E.g. OrderNumber in sales fact table

ID	OrderNumber		
1	A51273	..	..
2	72841	..	..
3	732GT5	..	..
4	624TR5K	..	..

TYPES OF FACT TABLES - TRANSACTIONAL

Transactional

- Most common
- Usually one row per line/event in a transaction
- Most detailed level
- The **grain must** (should) **be the same** for all rows
- Measures can usually be aggregated: “additive measure” (e.g. sum over sales amount)
- E.g. fact table for sales data

TYPES OF FACT TABLES – PERIODIC SNAPSHOT

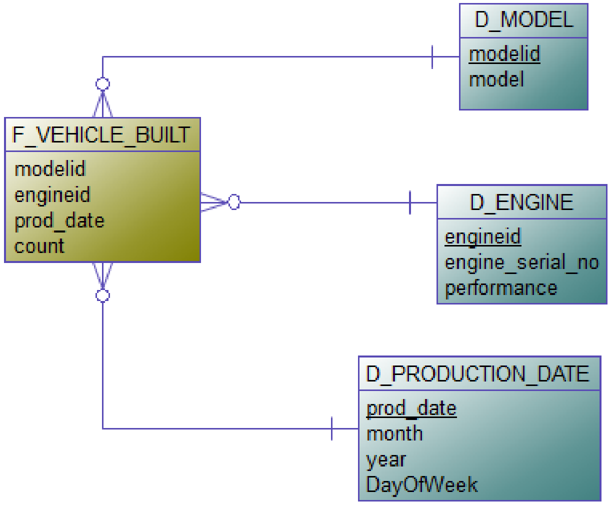
Periodic snapshots

- Picture of the time
- Often computed from transactional fact table, e.g. aggregated by month
- Measures can usually not be aggregated (e.g. sum over inventory does not make sense as inventory is already snapshot / sum for a day)
- The **grain must** (should) **be the same** for all rows
- E.g. fact table for inventory data (summed up for each day)

EXERCISE: QUERIES 1

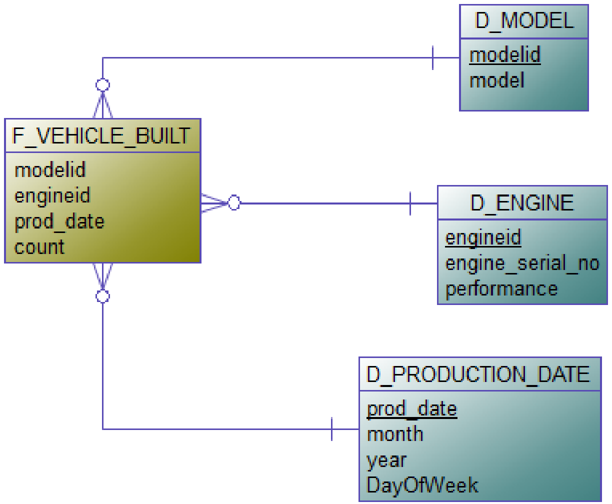
How many cabriolets (D_Model.model) have been Built in January and February 2016?

	Count
01/2016	
02/2016	



Assume SCD1 and no history in fact tables

EXERCISE: QUERIES 1



How many cabriolets (D_Model.model) have been Built in January and February 2016?

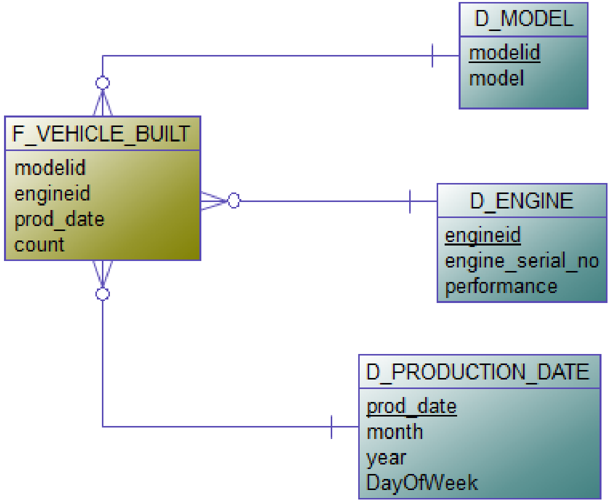
```
SELECT d.month, d.year, sum(f.count)
FROM   f_vehicle_built f
JOIN   d_model m on m.modelid = f.modelid
JOIN   d_production_date d on d.prod_date = f.prod_date
WHERE  m.model = 'Cabriolet'
AND    d.month IN (1, 2) AND d.year = 2016
GROUP BY d.month, d.year
```

EXERCISE: QUERIES 2

How many different models (D_Model.model) have

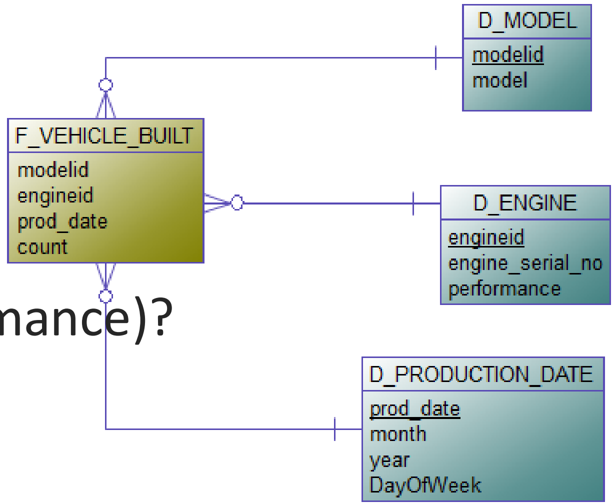
Currently a performance of 105PS (D_ENGINE.performance)?

Model	**Count**
Cabriolet	
SUV	
…	



Assume SCD1 and no history in fact tables

EXERCISE: QUERIES 2



How many different models (D_Model.model) have Currently a performance of 105PS (D_ENGINE.performance)?

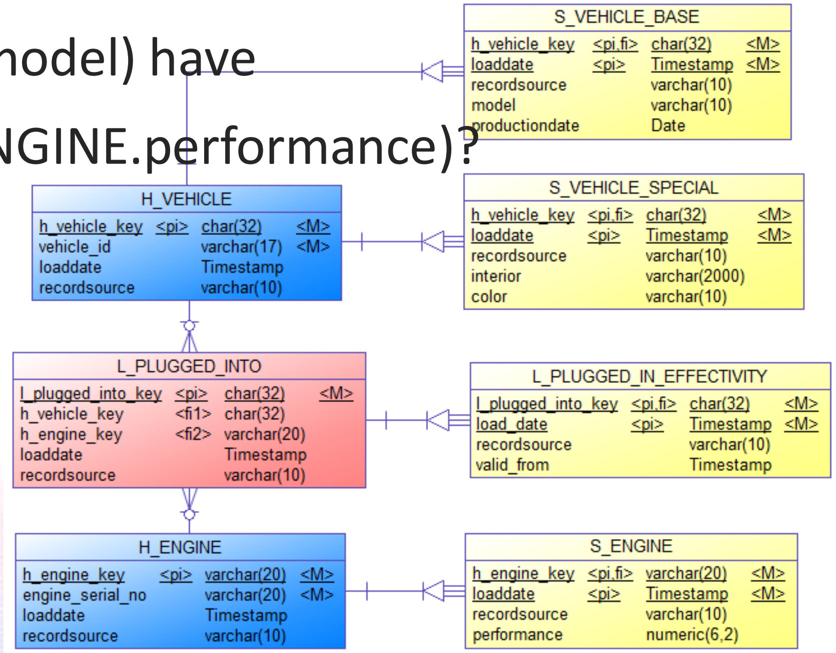
```
Select m.model, sum(f.count)
FROM   f_vehicle_built f
JOIN   d_model m on m.modelid = f.modelid
JOIN   d_engine e on e.engineid = engineid
WHERE  e.performance = 105
GROUP BY m.model
```

EXERCISE: QUERIES 3

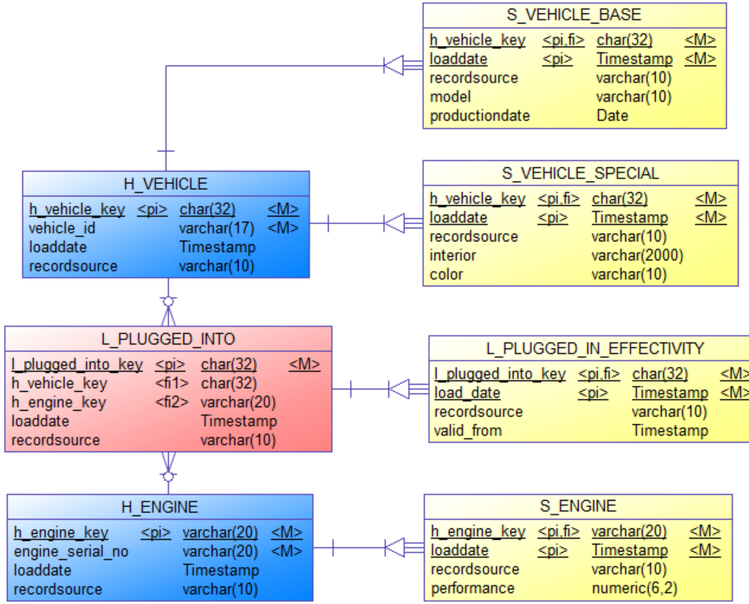
How many different models (D_Model.model) have

Currently a performance of 105PS (D_ENGINE.performance)?

Model	Count
Cabriolet	
SUV	
…	



EXERCISE: QUERIES 3



How many different models (D_Model.model) h

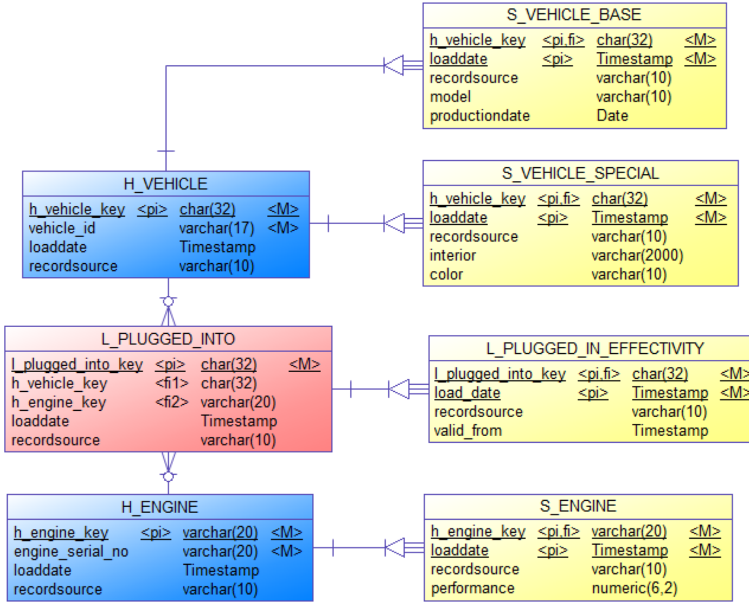
Currently a performance of 105PS (D_ENGINE.p

```
CREATE VIEW v_vehicle_sat as
SELECT h_vehicle_key, max(loaddate), model
FROM   s_vehicle_base
GROUP BY h_vehicle_key;
CREATE VIEW v_engine_sat as
SELECT h_engine_key, max(loaddate), performance
FROM   s_engine
GROUP BY h_engine_key;
```

EXERCISE: QUERIES 3

How many different models (D_Model.model) h

Currently a performance of 105PS (D_ENGINE.p




```
SELECT model, count(*)
FROM   v_vehicle_sat v
JOIN   l_plugged_into l ON l.h_vehicle_key = v.h_vehicle_key
JOIN   v_engine_sat e ON l.h_engine_key = e.h_engine_key
JOIN   s_engine s ON s.h_engine_key = e.h_engine_key
                 AND s.loaddate = e.loaddate
WHERE  s.performance = 105
GROUP  by model;
```

Many other solutions possible, e.g. using with clause instead of views or using window functions – all depending from DB vendor/version

MOLAP

Implementation types of dimensional models

Star Schema = Relational model (ROLAP) consists of

- Fact Tables
- Dimension Tables

Cube = Multidimensional model (MOLAP) consists of

- Edges = Attributes
- Cells = Measures (facts)

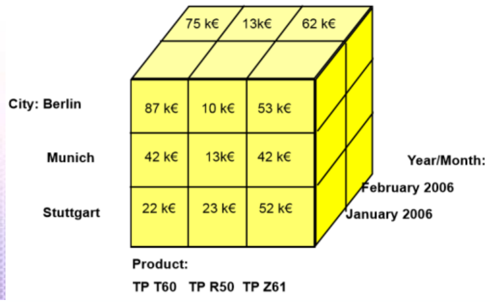
MULTIDIMENSIONAL DATA MODEL

Edges of a cube ("Dimension")

- Attributes like Product, Region, Time period (day, week, month, year)

Cells of a cube ("Measures")

- Key Figures (i.e. sales amount, profit) – "measures"
  - For every combination of attribute values one value of each key figure, e.g. Sales amount for product X in region y and time period z
  - Can be NULL and is stored as empty cell



MOLAP - MULTIDIMENSIONAL DATABASES

A database specially designed to handle the organization of data in multiple dimensions

- Good for DWH requirements only but not generally suited like a relational DBMS
- E.g. IBM Cognos TM1, Oracle Essbase, Microsoft Analysis Services, Oracle OLAP Option, IBM Cognos Powerplay

Holds data cells in blocks that constitute a virtual cube

Optimized to handle numeric data

- Aggregated totals often precalculated
- **Not intended for textual data**

MULTIDIMENSIONAL STORAGE

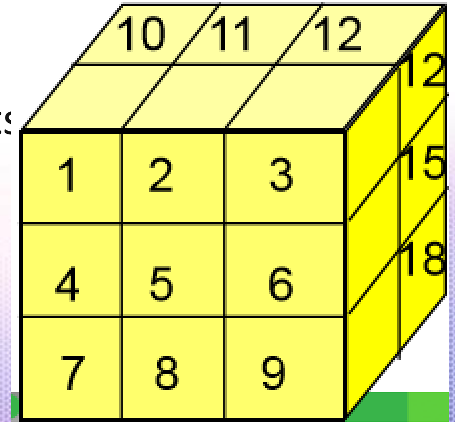
Linearization of the cells in a cube into a one-dimensional array

Memory amount: #(dim1) x #(dim2) x ... x #(dimN)

→ Depends on the **number of dimensions and their cardinality**, not on the number of facts

Example:

- Cube with 2 dimensions with 3 and 1 dimension with 2 elements
- Memory amount = size = 3*3*2 = 18 cells
- The numbers in the cube cells indicate the position in the array



EXAMPLE

Cube with 3 dimensions

- Product – 4 values – p1, p2, p3, p4
- Store – 3 values – s1, s2, s3
- Time (year) – 2 values - y1, y2

Number of cells in the cube: 4 x 3 x 2 = 24

p1,s1,y1	p2,s1,y1	p3,s1,y1	p4,s1,y1	p1,s2,y1	p2,s2,y1	p3,s2,y1	p4,s2,y1
1	2	3	4	5	6	7	8
p1,s3,y1	p2,s3,y1	p3,s3,y1	p4,s3,y1	p1,s1,y2	p2,s1,y2	p3,s1,y2	p4,s1,y2
9	10	11	12	13	14	15	16
p1,s2,y2	p2,s2,y2	p3,s2,y2	p4,s2,y2	p1,s3,y2	p2,s3,y2	p3,s3,y2	p4,s3,y2
17	18	19	20	21	22	23	24

EXAMPLE

Sales in year y2

p1,s1,y1	p2,s1,y1	p3,s1,y1	p4,s1,y1	p1,s2,y1	p2,s2,y1	p3,s2,y1	p4,s2,y1
1	2	3	4	5	6	7	8
p1,s3,y1	p2,s3,y1	p3,s3,y1	p4,s3,y1	p1,s1,y2	p2,s1,y2	p3,s1,y2	p4,s1,y2
9	10	11	12	13	14	15	16
p1,s2,y2	p2,s2,y2	p3,s2,y2	p4,s2,y2	p1,s3,y2	p2,s3,y2	p3,s3,y2	p4,s3,y2
17	18	19	20	21	22	23	24

EXAMPLE

Sales of store s1 in year y2

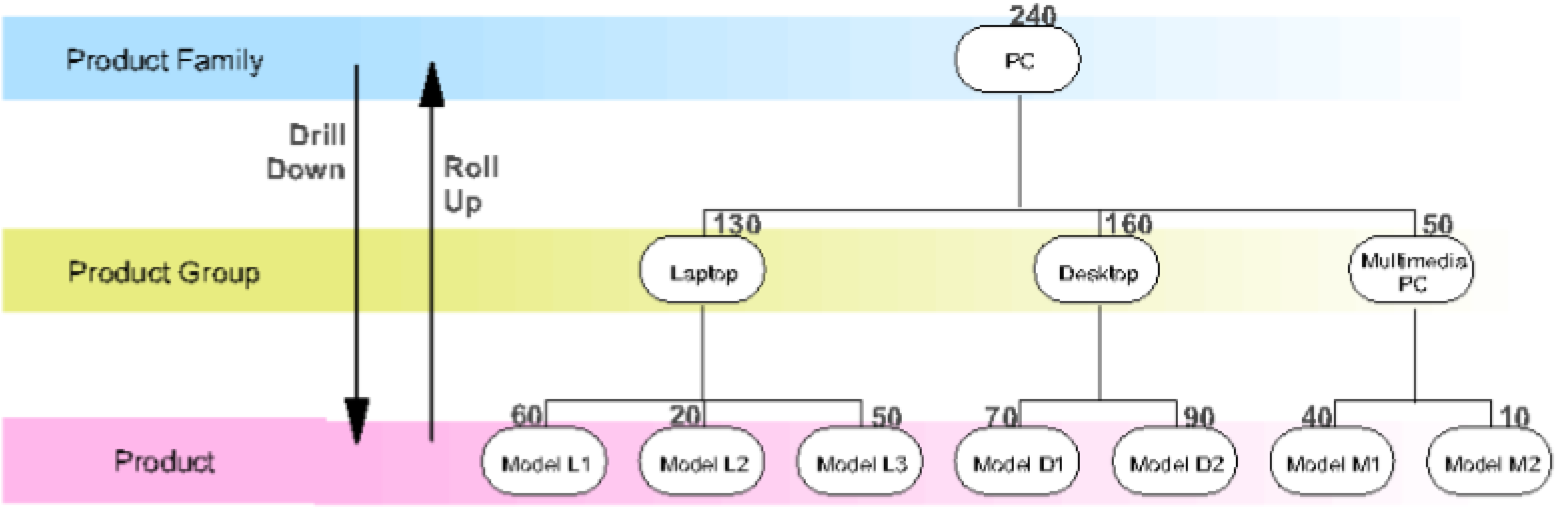
p1,s1,y1	p2,s1,y1	p3,s1,y1	p4,s1,y1	p1,s2,y1	p2,s2,y1	p3,s2,y1	p4,s2,y1
1	2	3	4	5	6	7	8
p1,s3,y1	p2,s3,y1	p3,s3,y1	p4,s3,y1	p1,s1,y2	p2,s1,y2	p3,s1,y2	p4,s1,y2
9	10	11	12	13	14	15	16
p1,s2,y2	p2,s2,y2	p3,s2,y2	p4,s2,y2	p1,s3,y2	p2,s3,y2	p3,s3,y2	p4,s3,y2
17	18	19	20	21	22	23	24

EXAMPLE

Sales of product p2 in year y1

p1,s1,y1	p2,s1,y1	p3,s1,y1	p4,s1,y1	p1,s2,y1	p2,s2,y1	p3,s2,y1	p4,s2,y1
1	2	3	4	5	6	7	8
p1,s3,y1	p2,s3,y1	p3,s3,y1	p4,s3,y1	p1,s1,y2	p2,s1,y2	p3,s1,y2	p4,s1,y2
9	10	11	12	13	14	15	16
p1,s2,y2	p2,s2,y2	p3,s2,y2	p4,s2,y2	p1,s3,y2	p2,s3,y2	p3,s3,y2	p4,s3,y2
17	18	19	20	21	22	23	24

ROLL-UP & DRILL-DOWN



MDX - OLAP QUERY LANGUAGE

ROLAP = SQL is standard language

MOLAP = MDX - Multidimensional Expressions

- De-facto industry standard developed by Microsoft
- Very complex
- SQL like syntax
- Language elements
  - Scalar – data type „string“ or „number“
  - Dimension, Hierarchy, Level, Member
  - …

MDX SAMPLE QUERY

	Store Sales
2002	95863,66
2003	99764,01

```
SELECT
{ [Measures].[Store Sales] } ON COLUMNS,
{ [Date].[2002], [Date].[2003] } ON ROWS
FROM Sales
WHERE ( [Store].[USA].[CA] )
```

This query defines the following result set information:

- The SELECT clause sets the query axes as the Store Sales (amount) member and the 2002 and 2003 members of the Date dimension
- The FROM clause indicates that the data source is the Sales cube
- The WHERE clause defines the "slicer axis" for member California of Store dimension

MOLAP - ROLAP

	MOLAP	ROLAP
Database type	Multidimensional	Relational
Data storage	Special storage engines for cube data	Star schema – special relational data model
Size	100s of Gigabytes	10s of Terabytes
Query language	MDX	SQL

MOLAP - ROLAP

	MOLAP	ROLAP
Advantages	• special database products optimized for multidimensional analysis • short response times, e.g. no joins • suitable storage schema and query processing for multidimensional data	• can use existing, well established DBMS • easy data import, update • user access, backup, security mechanisms from DBMS can be used
Disadvantages	• problems with sparsity (ratio occupied / not occupied cells): "null" is stored in a field with same length as any value • limited data volume: 5-6 dimensions • cube data read-only accessible only for end users • expensive update operation	• Complex SQL queries for processing OLAP requests → longer response times (solution: Materialized Views and In-memory columnar databases)

HOLAP – HYBRID OLAP

Combines the advantages or ROLAP and MOLAP

Relational DBMS for storage of sparse, historic data

- Data of highest granularity level

Multidimensional DBMS for efficient storage of dense data cubes

- Multidimensional cache for aggregated totals

Complex architecture and maintenance processes

No uniform OLAP query processing

EXERCISE: OLAP

The following is a data model used by a supermarket chain to analyze their business:

EXERCISE: OLAP

With each transaction, an average of 20 different articles are bought.

The data warehouse collects sales transactions data over 2 years.

There are 1000 stores with 2000 transactions per store and day.

Questions:

- 1. What are the columns of the ROLAP fact table?
- 2. How many records are stored in the fact table?
- 3. What is the size of the cube (number of cells) that stores the aggregated values at the most detailed level?
- 4. Compute the respective cube sizes for the other 3 (higher) hierarchy levels.

EXERCISE: OLAP

1. What are the columns of the ROLAP fact table?

- Trans. No. (FK to dimension)
- Date (FK to dimension)
- Location (FK to dimension)
- Article (FK to dimension)
- No of articles (measure) and Article Price (measure)

2. How many records are stored in the fact table?

- One record per transaction and article (with quantity and price)
- 2 years * 365 days/year * 1000 stores * 2000 transactions/(store*day)* 20 articles/transaction = 29.200.000.000 articles/records

EXERCISE: OLAP

3. What is the size of the cube (number of cells) that stores the aggregated values at the most detailed level?

- 2 years * 365 [days]/year * 2000 [transactions] * 1000 [stores] * 50000 [articles] = 73.000.000.000.000 cells

4. Compute the respective cube sizes for the other 3 hierarchy levels.

- Level 2: 2 years * 12 [months]/year * 500 [cities] * 2000 [product groups] = 24.000.000 cells
- Level 3: 2 years * 4 [quarters]/year * 20 [regions] * 200 [product categories] = 32.000 cells
- Level 4: 2 [years] * 5 [regions] * 10 [product departments] = 100 cells

SUMMARY

- Data modeling in the Core Warehouse Layer
  - Choices like Data Vault
- Data modeling in the Mart Layer
  - Dimensional Modeling
    - ROLAP (Star Schema with fact and dimension tables)
    - MOLAP (Cubes)

EXERCISE - RECAPTURE DATA MODELING

- Recapture data modeling topics
- Which topics do you remember or do you find important?
- Write down 1-2 topics on stick-it cards.

THANK YOU

Daimler TSS

Daimler TSS GmbH
Wilhelm-Runge-Straße 11, 89081 Ulm / Telefon +49 731 505-06 / Fax +49 731 505-65 99
tss@daimler.com / Internet: www.daimler-tss.com/ Intranet-Portal-Code: @TSS
Domicile and Court of Registry: Ulm / HRB-Nr.: 3844 / Management: Christoph Röger (CEO), Steffen Bäuerle

OLAP – 12 CRITERIA BY CODD

OnLine Analytical Processing

- Term introduced by E. Codd in 1993 in a white paper for Arbor Essbase
- 12 criteria for OLAP systems like
  - Multi-dimensionality
  - Transparency
  - Constant response-times
  - Multi-user support
  - Flexible definition of reports
  - No limits on dimensions and hierarchy levels

OLAP – FASMI CRITERIA

FASMI – Fast Analysis of Shared Multidimensional Information

Criteria by Pendse/Creeth (1995)

- **F**ast
  - maximum response time for regular queries 5 seconds and complex queries not more 20 seconds
- **A**nalysis
  - intuitive analysis, easy/no programming
  - flexible: queries may contain arbitrary computations

OLAP – FASMI CRITERIA

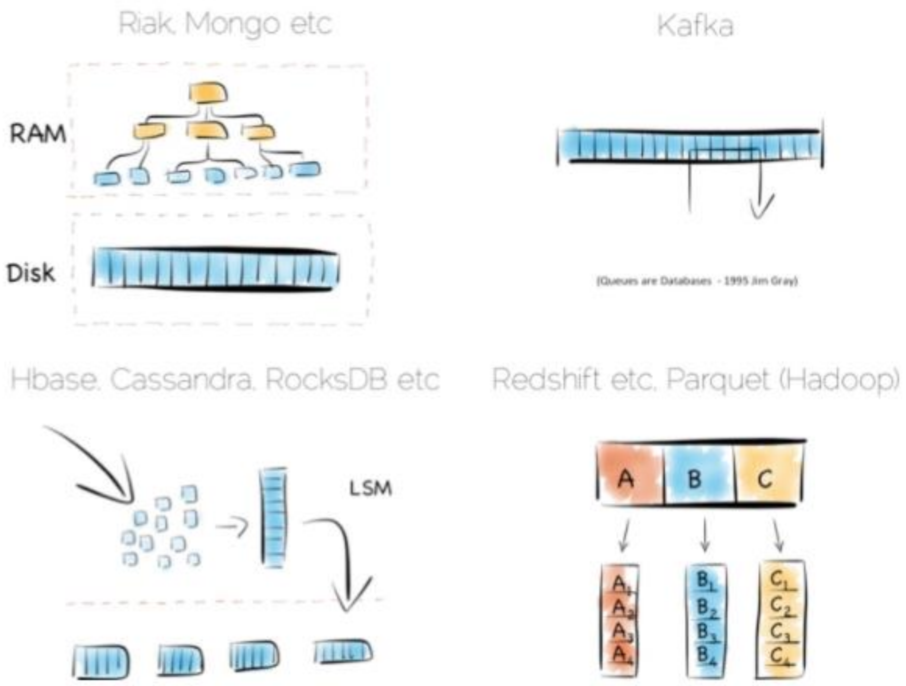
- **S**hared
  - Multi user capable: Shared usage and access control
- **M**ultidimensional
  - Multidimenional view on the data
    - regardless of the underlying data model
    - Full support of hierarchies
- **I**nformation
  - User must be able to get all data without restrictions by the used OLAP system, no restriction in regards to scalability

ELEMENTS OF SCALE: COMPOSING AND SCALING DATA PLATFORMS (BEN STOPFORD)

- Sequential operations are best
- Sequential operations can be predicted
- Random operations are the main challenge
- Append-only journal leads to sequential IO
- But what about updates (in place)?
    - Indexes speed up read random IO read performance but not random IO write performance

Source: http://www.benstopford.com/2015/04/28/elements-of-scale-composing-and-scaling-data-platforms/

ELEMENTS OF SCALE: COMPOSING AND SCALING DATA PLATFORMS (BEN STOPFORD)



Source: http://www.benstopford.com/2015/04/28/elements-of-scale-composing-and-scaling-data-platforms/

MULTIDIMENSIONAL OPERATIONS - SELECTION

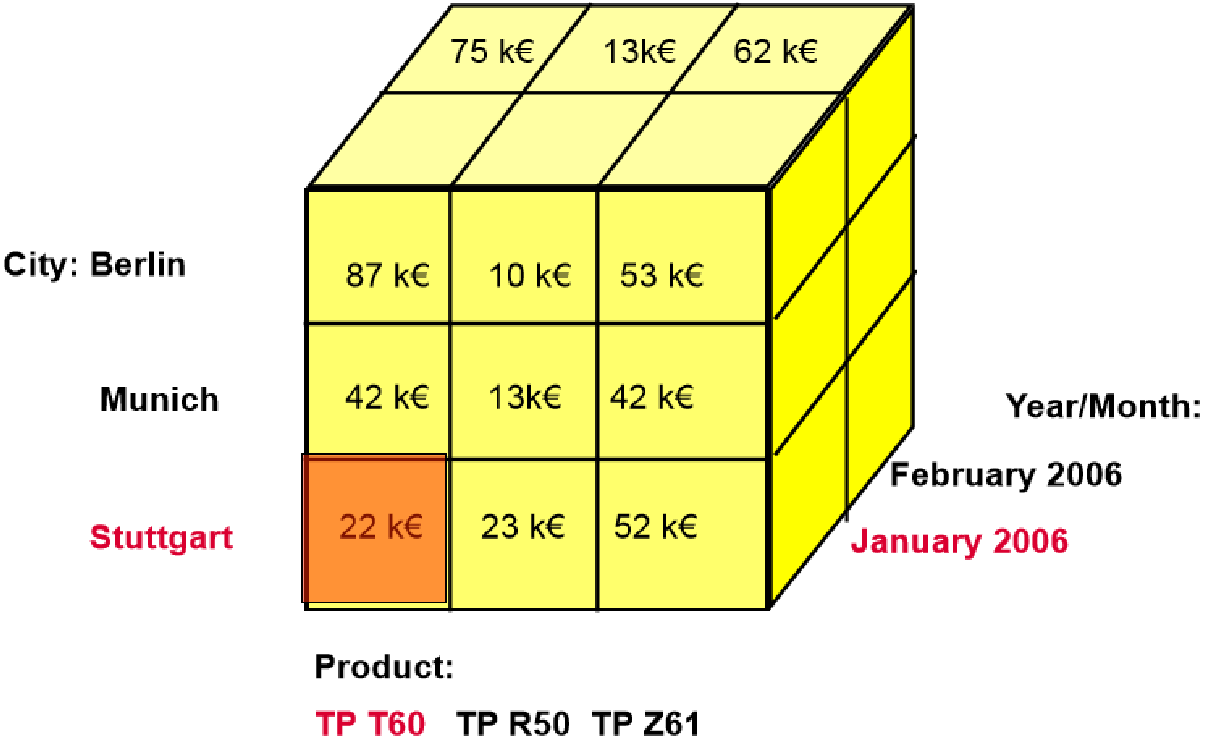
Selection

Definition of a filter

Select **data of a single cell** with a condition **for each dimension**

- For instance:
  - time = 'January 2006'
  - location = 'Stuttgart'
  - product = ‘ThinkPad T60'

EXAMPLE SELECTION



MULTIDIMENSIONAL OPERATIONS - SLICE

Slice

Definition of a filter

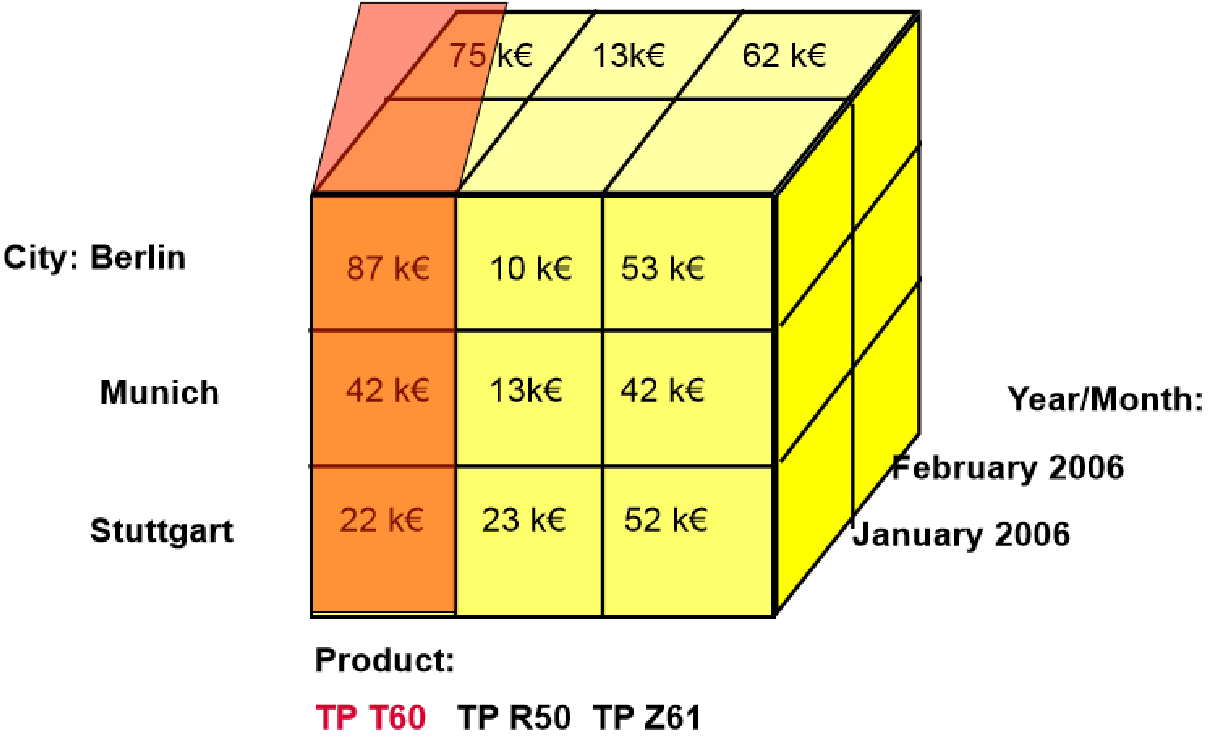
Condition for one single dimension

Select a **new cube with one fewer dimension**

For instance

- Product = ‘ThinkPad T60'

EXAMPLE SLICE



MULTIDIMENSIONAL OPERATIONS - DICE

Dice

Definition of intervals/sets as filter

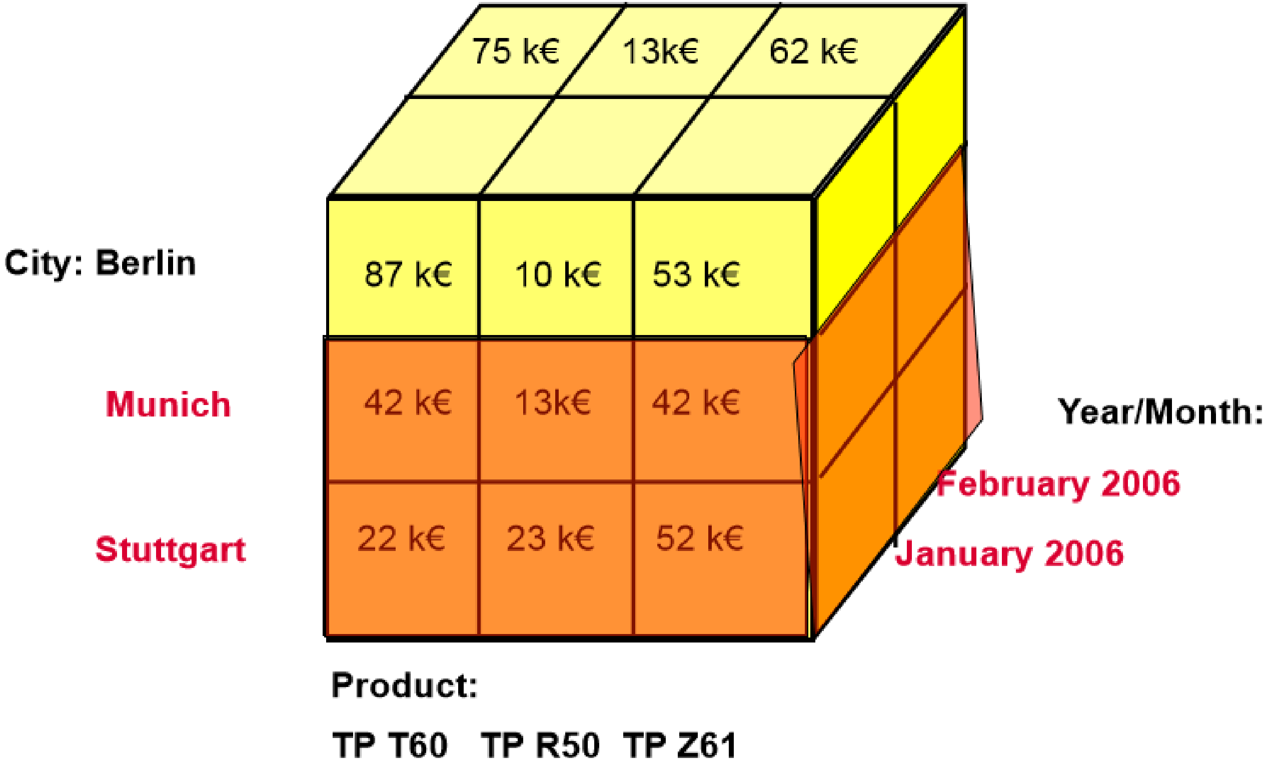
Pick **specific values of multiple dimensions**

Select a **smaller cube**

Conditions for instance

- time = 1st quarter (January, February, March)
- location = region south (Stuttgart, Frankfurt, Munich)

EXAMPLE DICE



MULTIDIMENSIONAL OPERATIONS – ROTATE/PIVOT

Rotate/Pivot

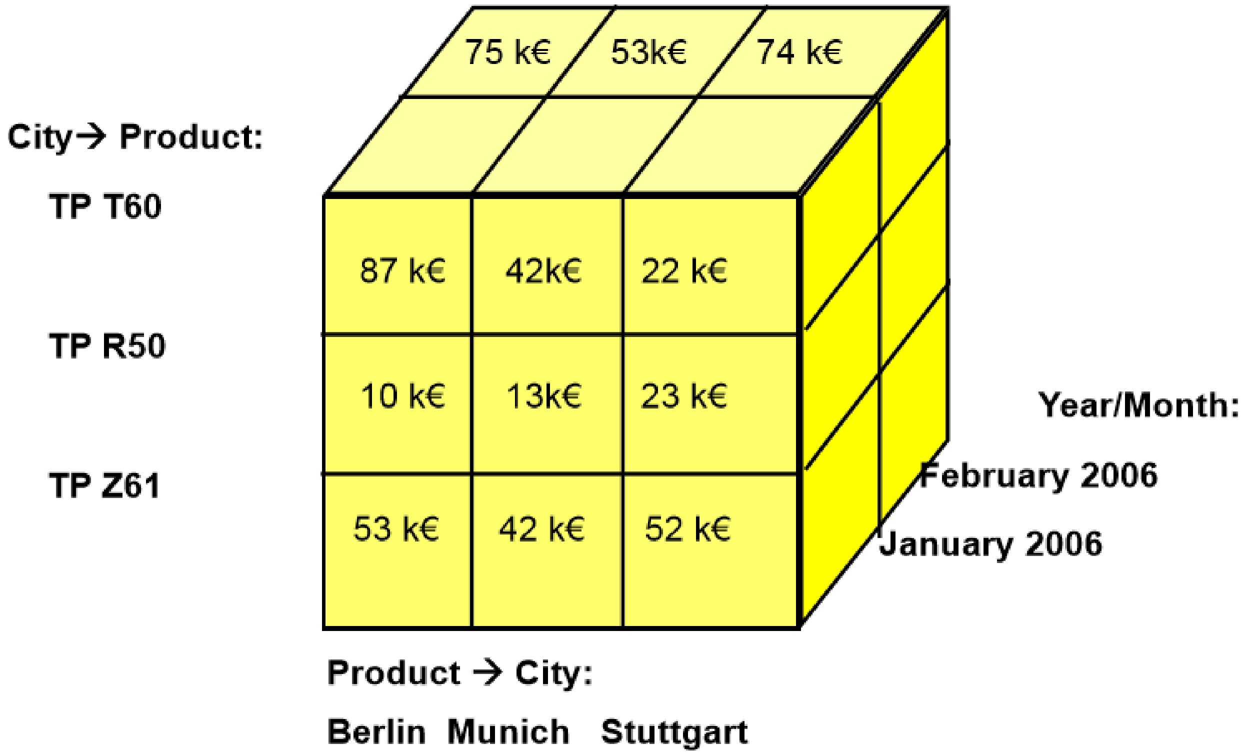
Rotate cube along its axes

Get **different view** on data cube

# of views on cube = (# of dimensions)!

- 2 dimensions, 2 views (2! = 2*1)
- 3 dimensions, 6 views (3! = 3*2*1)
- 4 dimensions, 24 views (4! = 4*3*2*1)
- ...

EXAMPLE ROTATE/PIVOT



MULTIDIMENSIONAL OPERATIONS – ROLL-UP/DRILL-DOWN

Roll-up & Drill-down

Prerequisites:

- Hierarchies defined
- Aggregated data for all hierarchy levels available

Roll up: change hierarchy level "upwards":

- get **less detailed data (= higher aggregation)**

Drill down: change hierarchy level "downwards":

- get **more detailed data (= lower aggregation)**

TYPES OF FACT TABLES - ACCUMULATING

Accumulating snapshots

Shows activity of a process/event over time

The data is not complete at the beginning and is updated as soon as new data arrived (e.g. delivery date can be unknown at the beginning)

The **grain must** (should) **be the same** for all rows

E.g. fact table for processing an order