

# Datawarehousing and Analytics

Exercises 1-7

Holger Schwarz  
Universität Stuttgart

winter term 2015/2016

# Exercise 1

## Data Warehouse Architecture

# Case Study

- Mountain States Health Alliance (MSHA)

## Franklin Woods Community Hospital (FWCH)



A 80-bed hospital offering a full array of primary care and some specialty services

## SYNERGY Laboratories (SYN)



full-service laboratory (e.g. for blood count), comprised of the laboratories within Mountain States Health Alliance

## Norton Community Hospital (NCH)



A 129-bed, acute-care facility that has been serving Southwest Virginia and Southeastern Kentucky

## James H. and Cecile C. Quillen Rehabilitation Hospital (QRH)



A 47-bed comprehensive inpatient rehab hospital

## Mountain States Imaging Center (MSIC)

Provides a wide array of high-quality diagnostic imaging services such as MRI, CT scan, and X-Ray



# Case Study

- In order to make its health care system more efficient, 29 counties in Tennessee, Virginia, Kentucky and North Carolina decided to establish the **Mountain States Health Alliance (MSHA)** which includes among others the following medical institutions of the region:
  - The **Franklin Woods Community Hospital (FWCH)** in Johnson City: A 80-bed hospital offering a full array of primary care and some specialty services.
  - The **James H. and Cecile C. Quillen Rehabilitation Hospital (QRH)** in Johnson City: A 47-bed comprehensive inpatient rehab hospital.
  - The **Norton Community Hospital (NCH)**, located in Norton, Virginia: A 129-bed, acute-care facility that has been serving Southwest Virginia and Southeastern Kentucky since 1949.
  - **SYNERGY Laboratories (SYN)** is a full-service laboratory (e.g. for blood count), comprised of the laboratories within Mountain States Health Alliance.
  - The **Mountain States Imaging Center (MSIC)** provides a wide array of high-quality diagnostic imaging services such as MRI, CT scan, and X-Ray.

# Case Study

Institution	Data Set	Notes
FWCH	department and beds	Information on beds per department and when they were occupied
	patients	Patients with names, health status, insurance information, ...
	employees	Staff of the hospital
	doctors	Doctors
	stock	Used drugs and their stock level
	procedures	Treatments of patients with date, treatment, doctor, etc.
QRH	patients	Patients with names, health history, insurance information, ...
	departments	Subject of department, available equipment
	bed	Beds and when they were occupied
	treatments	Treatments of patients with date, treatment, staff, etc.
	payments	Billing information, open accounts, ...
	staff	Nurses and other staff of the hospital
	doctors	Doctors and their specialization
MSIC	drugs	Stock level of drugs
	treatments	Protocol and results of MRI, CT scan, and X-Ray, etc.
	schedule	Planned treatments
SYN	treatments	Blood counts etc. with time, patient, result, etc.
Pharmacy	Inventory	Drugs and their stock level
	Orders	Orders and deliveries by/to hospitals and other institutions

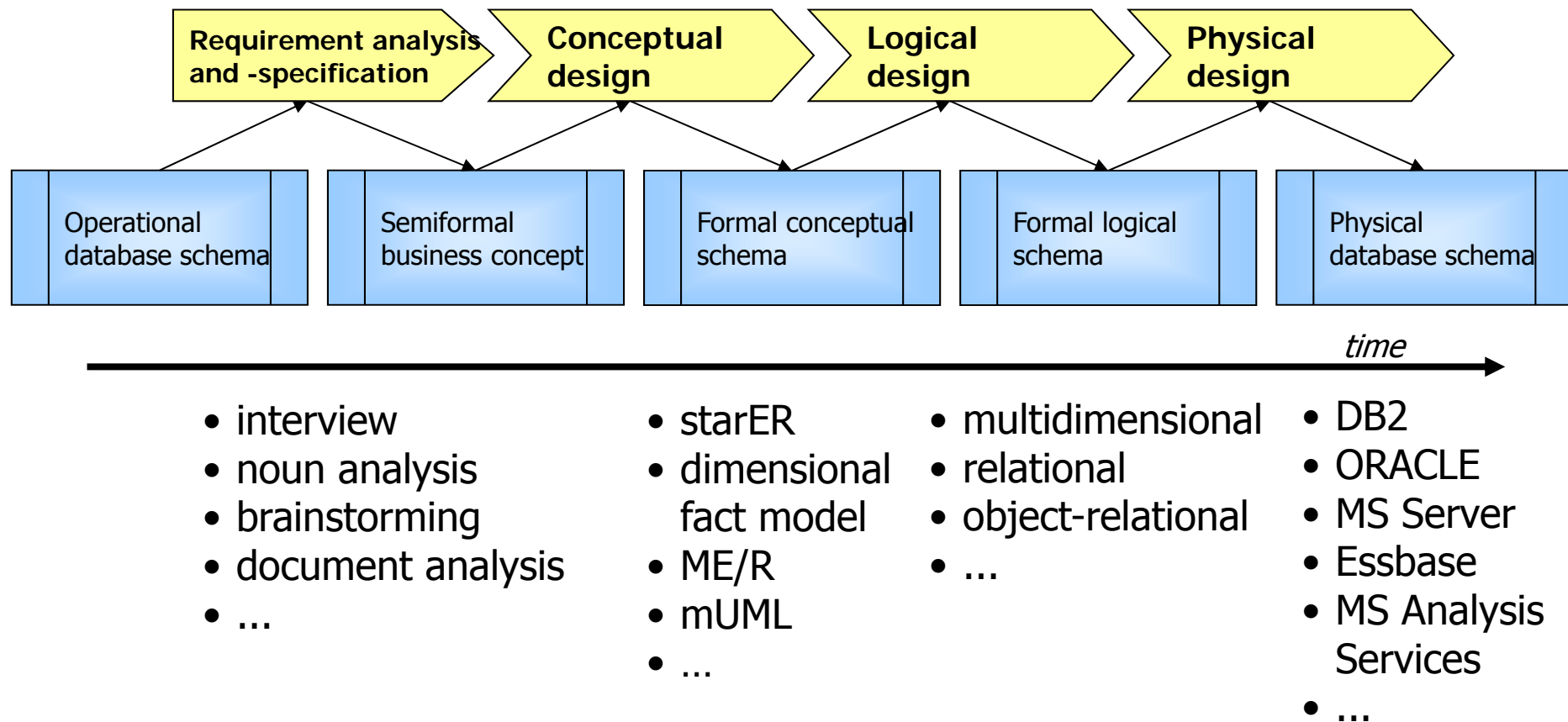
# Case Study

- MSHA also has a close cooperation with the **Emory University School of Medicine** (EUSM) in Atlanta. The university runs various projects on Rehabilitation Medicine and wants to use data from MSHA for its studies.
- Making all relevant information available to doctors and nurses in all hospitals as well as to the management of all institutions in MSHA is important to achieve an effective and efficient health care system. In particular, the management wants to establish an **early warning system** for the following events:
  - (1) A certain type of drugs is running out of stock in the hospitals of MSHA,
  - (2) a critical level of MRSA (Methicillin-resistant Staphylococcus aureus) contamination is reached.
- In a first step, the MSHA decided to leave the IT systems of each of these institutions as they are. Nevertheless, an integrated view on all the information related to patients, diagnoses and treatments is needed. Thus a data warehouse project is initiated. Discuss the following questions related to the motivation for and the architecture of a data warehouse system providing this integrated view.

## Task 2: Information Needs

- Try to find **examples for questions** that might be relevant for decisions of
  - a) a doctor at the FWCH,
  - b) a doctor at the MSIC,
  - c) the manager of the QRH,
  - d) the general manager of MSHA, and
  - e) the pharmacy supplying drugs to all members of MSHA.
- Which data of which of the institutions mentioned above is needed to answer these questions?

# Data Warehouse Design Process





- Sample questions for a doctor at the FWCH

Question	Data (non-exhaustive)
Current/former patients of FWCH with a diagnosis similar to patient A	FWCH:patients, FWCH:procedures
Number of currently unoccupied beds (per department)	FWCH:department and beds
Patient profiles (from all hospitals)	*:patients
Which colleague may I contact to discuss a certain diagnosis?	*:doctors
Schedule of imaging center (when is the next possible appointment for CT imaging?)	MSIC:schedule
which drugs are immediately available at the FWCH?	FWCH:stock, Pharmacy:Inventory,
What is the current level of MRSA contamination in FWCH?	FWCH:patients, SYN:treatments

- Sample questions for a doctor at the MSIC (Mountain States Imaging Center)

Question	Data (non-exhaustive)
Which MRIs and CT scans of the head (in the last quarter) showed a certain problem	MSIC: treatments
Schedule of imaging center (when is the next possible appointment for CT imaging?)	MSIC: schedule
Which colleague may I contact to discuss a certain diagnosis?	*:doctors

- Sample questions of the manager of the QRH (James H. and Cecile C. Quillen Rehabilitation Hospital)

Question	Data (non-exhaustive)
Available beds (locally and in other hospitals)	QRH:departments, QRH:beds, FWCH:beds and departments
Overview on stock level of drugs (locally and in other hospitals)	QRH:drugs, FWCH:stock, Pahrmary:Inventory
Staff information, e.g., number of sick leaves of doctors and nurses	QRH:staff, QRH:doctors
Local statistics: patient per doctor ratio, patient per nurse ratio	QRH:staff, QRH:doctors, QRH:patients
Financial volume of treatments not paid by health insurances so far	QRH: treatments, QRH:payments
Average duration of stay of patients in certain rehab programs	QRH:patients, QRH:departments, QRH:beds

- Sample questions of the general manager of MSHA (Mountain States Health Alliance)

Question	Data (non-exhaustive)
Income level	QRH:staff, *:employees, *:doctors
Statistics: patient per doctor ratio, patient per nurse ratio	QRH:staff, *:employees, *:doctors, *:patients
Load of imaging center	MSIC:schedule
Average utilization of beds per hospital and month	*:beds
Average utilization of departments per hospital	*:beds, *:departments
Number and duration of sick leaves of doctors and nurses per hospital and in general	QRH:staff, *:employees, *:doctors

- Sample questions of pharmacy supplying drugs to all members of MSHA (Mountain States Health Alliance)

Question	Data
Stock level of drugs in each hospital	*:stock
Drugs typically used per hospital, find suggestions to harmonise the usage of drugs	*:treatments, FWCH:procedures, Pharmacy: orders

## Task 3: Issues of Information Integration

- Think of **potential problems** that could arise when integrating data of all members of MSHA. Think of various dimensions of data sources and possible issues with data quality.

# Dimensions of Data Sources

origin

- internal vs. external data

time

- current vs. historic data

usage

- data vs. metadata

type

- number, string, time, graphic, audio, video, ...  
numeric, alphanumeric, boolean, binary, ...

character set

- ASCII, EBCDIC, UNICODE, ...

orientation

- left to right, right to left, top-down

confidentiality

- strictly confidential, confidential, public, ...

# Data Quality

consistency	<ul style="list-style-type: none"><li>• Are there contradictions in data and/or metadata?</li></ul>
correctness	<ul style="list-style-type: none"><li>• Do data and metadata provide an exact picture of the reality?</li></ul>
completeness	<ul style="list-style-type: none"><li>• Are there missing attributes or values?</li></ul>
exactness	<ul style="list-style-type: none"><li>• Are exact numeric values available?</li><li>• Are different objects identifiable? Homonyms?</li></ul>
reliability	<ul style="list-style-type: none"><li>• Is there a Standard Operating Procedure (SOP) that describes the provision of source data?</li></ul>
understandability	<ul style="list-style-type: none"><li>• Does a description for the data and coded values exist?</li></ul>
relevance	<ul style="list-style-type: none"><li>• Does the data contribute to the purpose of the data warehouse?</li></ul>



### consistency

- Same system to encode diseases? There might be exceptions being treated differently
- Is patient information consistent in various hospitals?

### correctness

- Correctness of diagnosis data

### completeness

- Missing address information
- Are all treatments and diagnoses covered by the data?

### exactness

- examinations/treatments may be recorded on a daily basis or the exact time may be available (granularity)
- various units: e.g. single pills or entire drug packages

### reliability

- How is the data about CT scans etc. entered into the system (by hand, automatically)?

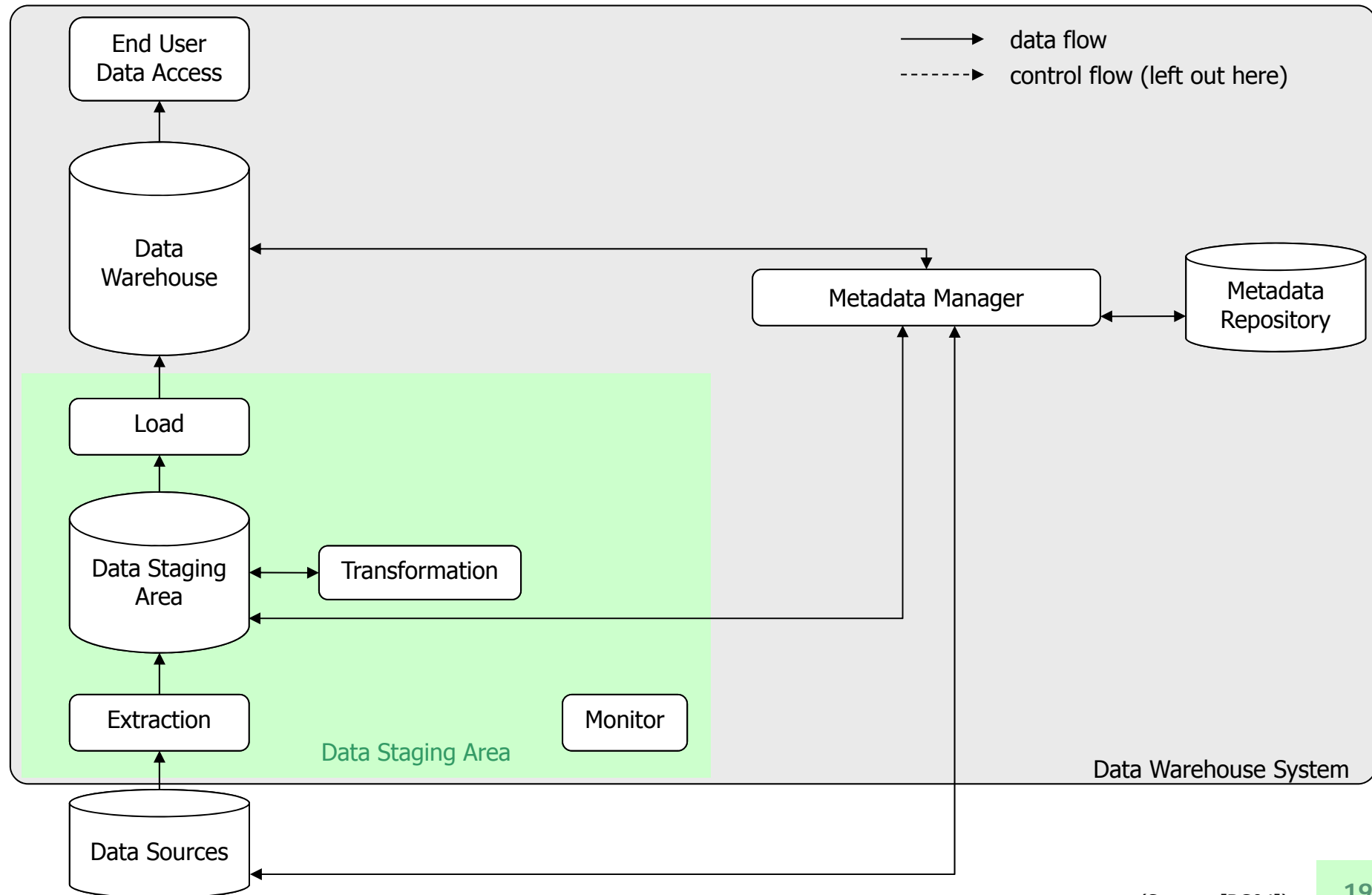
### understandability

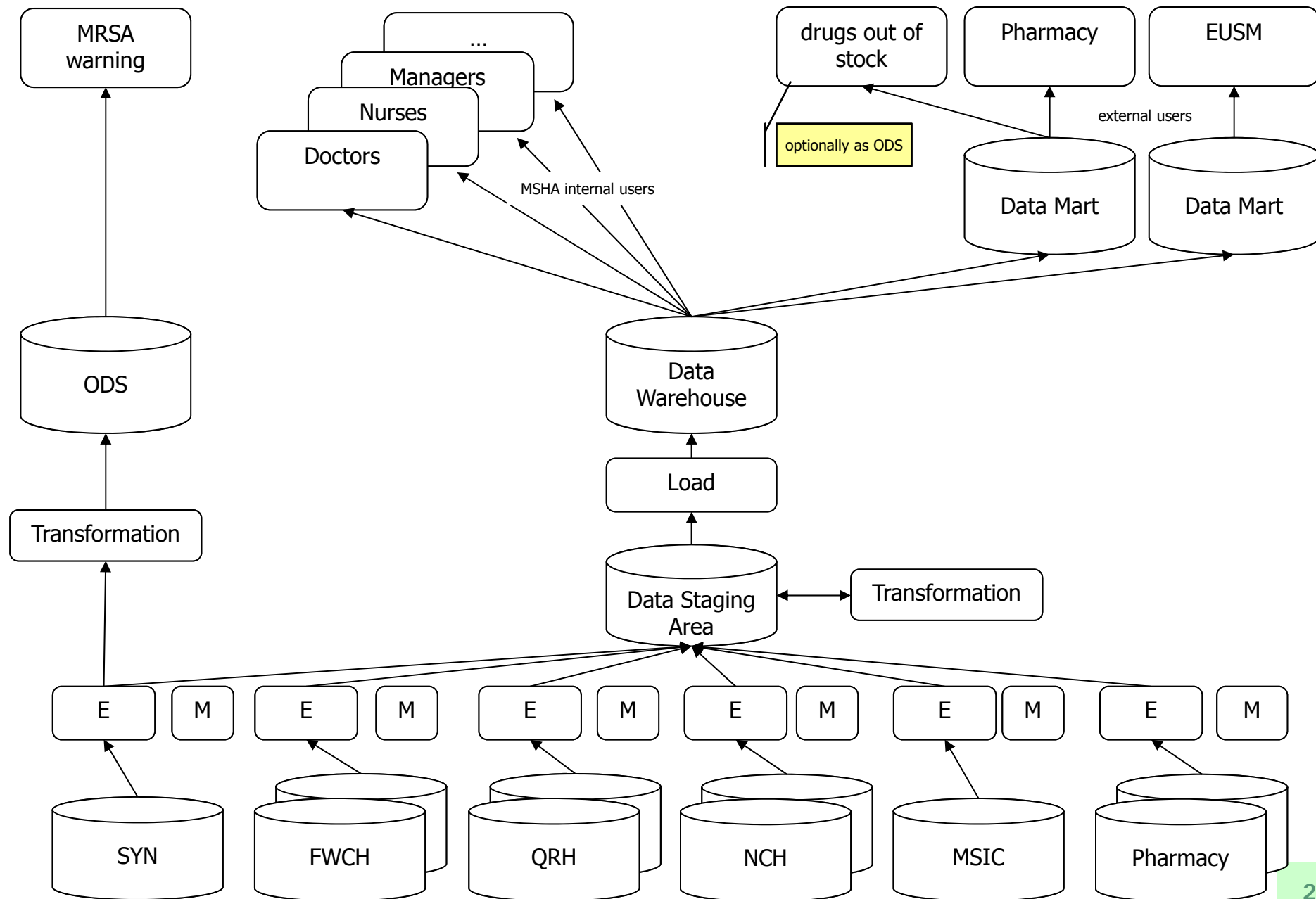
- working hours are very often encoded (e.g. M for morning shift); this may differ from hospital to hospital
- various units: e.g. duration in seconds/minutes/...

## Task 4: Data Warehouse Architecture

- Sketch the **architecture** of a Data Warehouse for MSHA (focus on data flow). Discuss possible applications of Data Marts and Operational Data Stores

# Architecture





## Task 5: Data Marts

- What are the advantages and disadvantages of using data marts?
- Compare them to those of a central data warehouse architecture.
- How do **dependent** data marts differ from **independent** data marts? What are appropriate usage scenarios for each of these types of data marts?
- Discuss possible applications of Data Marts in the MSHA data warehouse.

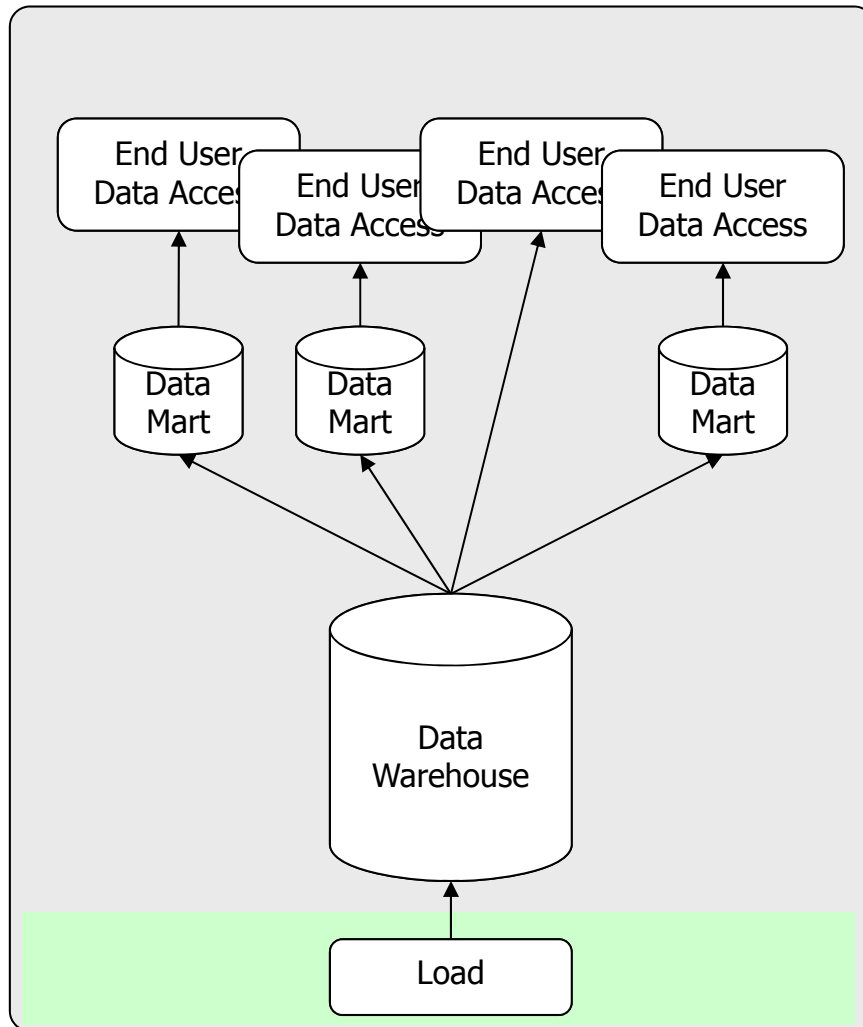
## Task 5: Data Marts

- Advantages of a central data warehouse
  - Only one data model necessary
  - Maintain only one data store
  - All issues of information integration are addressed in the ETL processes for the central data warehouse
- Disadvantages of a central data warehouse
  - Performance issues due to huge amount of data
  - There may be scalability issues
  - Complexity issues: complex data model, many potentially complex ETL processes

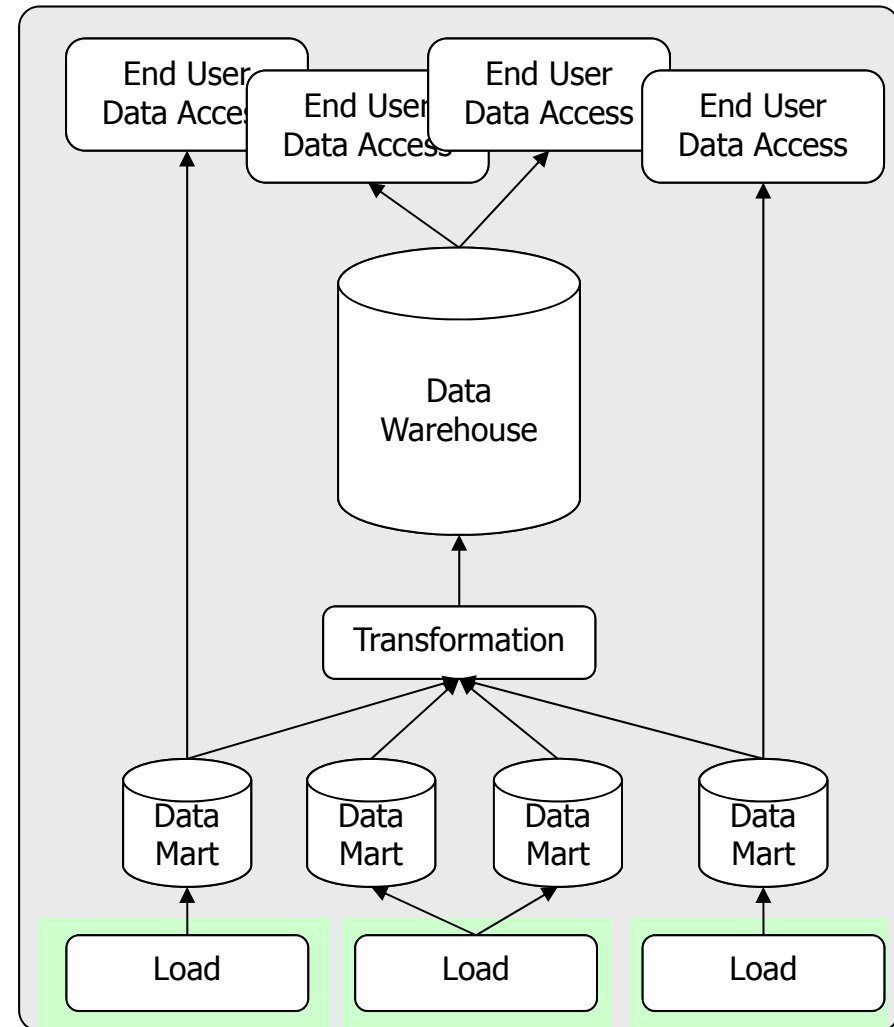
## Task 5: Data Marts

- Motivation for data marts
  - less complex data models in data marts
  - reduced data volume
  - efficient query processing due to data volume and aggregated data
  - supports load balancing
  - provide specific data for an organizational unit/user group
  - provide aggregated data
  - data protection: organizational unit/user group only gets access to a subset of data
  - data mart independent from the ETL processes of the central data warehouse

# Data Marts



dependent data marts



independent data marts



## Task 5: Data Marts

- Dependent data mart
  - build central data warehouse first
  - derive data mart from the central data warehouse
  - data mart is basically a view on the data warehouse
  - central data warehouse and data mart provide consistent information
  - more suitable for a data warehouse project in an existing organization
- Independent data mart
  - build data marts first
  - set up additional ETL processes that consolidate data into the central data warehouse (to support more comprehensive analysis)
  - ETL processes on various layers of the overall system
  - Inconsistent analysis possible
  - more suitable for data warehouses in the context of mergers

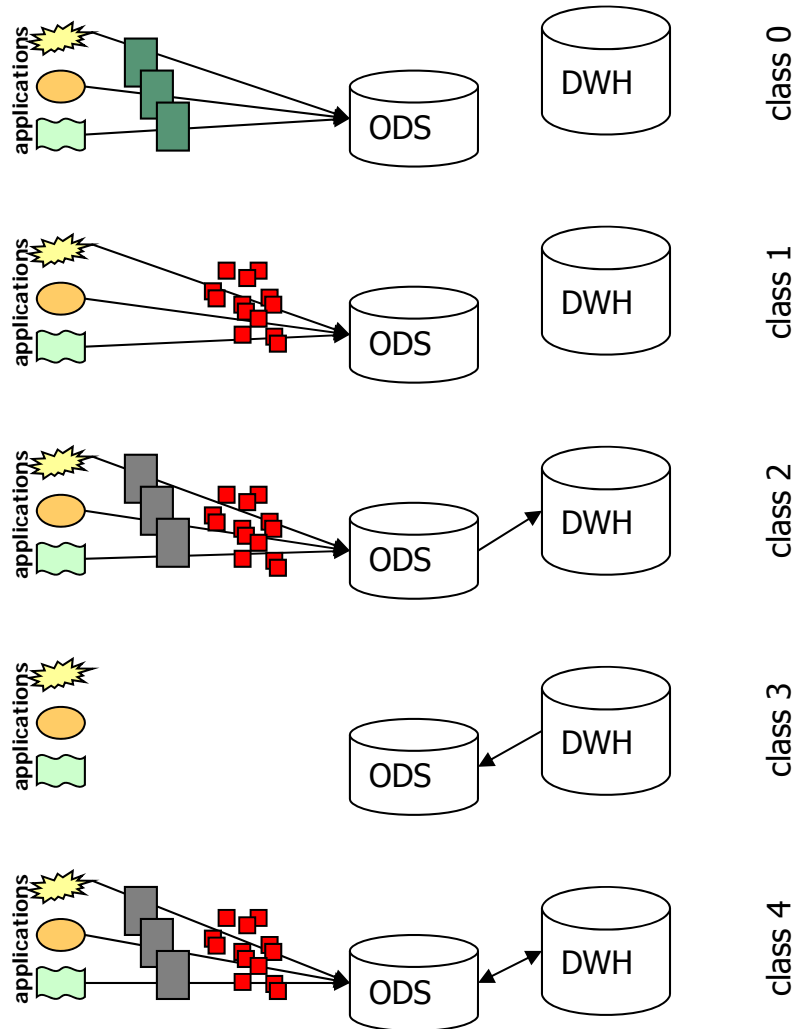
## Task 6: Operational Data Store

- Discuss possible applications of **Operation Data Stores (ODS)** in the MSHA data warehouse.

## Task 6: Operational Data Store

- Many interpretations of this term (see lecture slides)
- Characteristics (in comparison to a typical data warehouse)
  - more frequent updates
  - less complex transformations
  - more detailed granularity
  - basis for (short-term) operational decisions

# Classes of Operational Data Stores



- Tables are copied from the operational environment
- Transactions are moved to the ODS in an immediate manner (range of one to two seconds)
- Activities in the operational environment are stored, integrated, and forwarded to the ODS
- ODS is fed aggregated analytical data from the data warehouse
- Combination of integrated data from the operational environment and aggregated data from the analytical environment

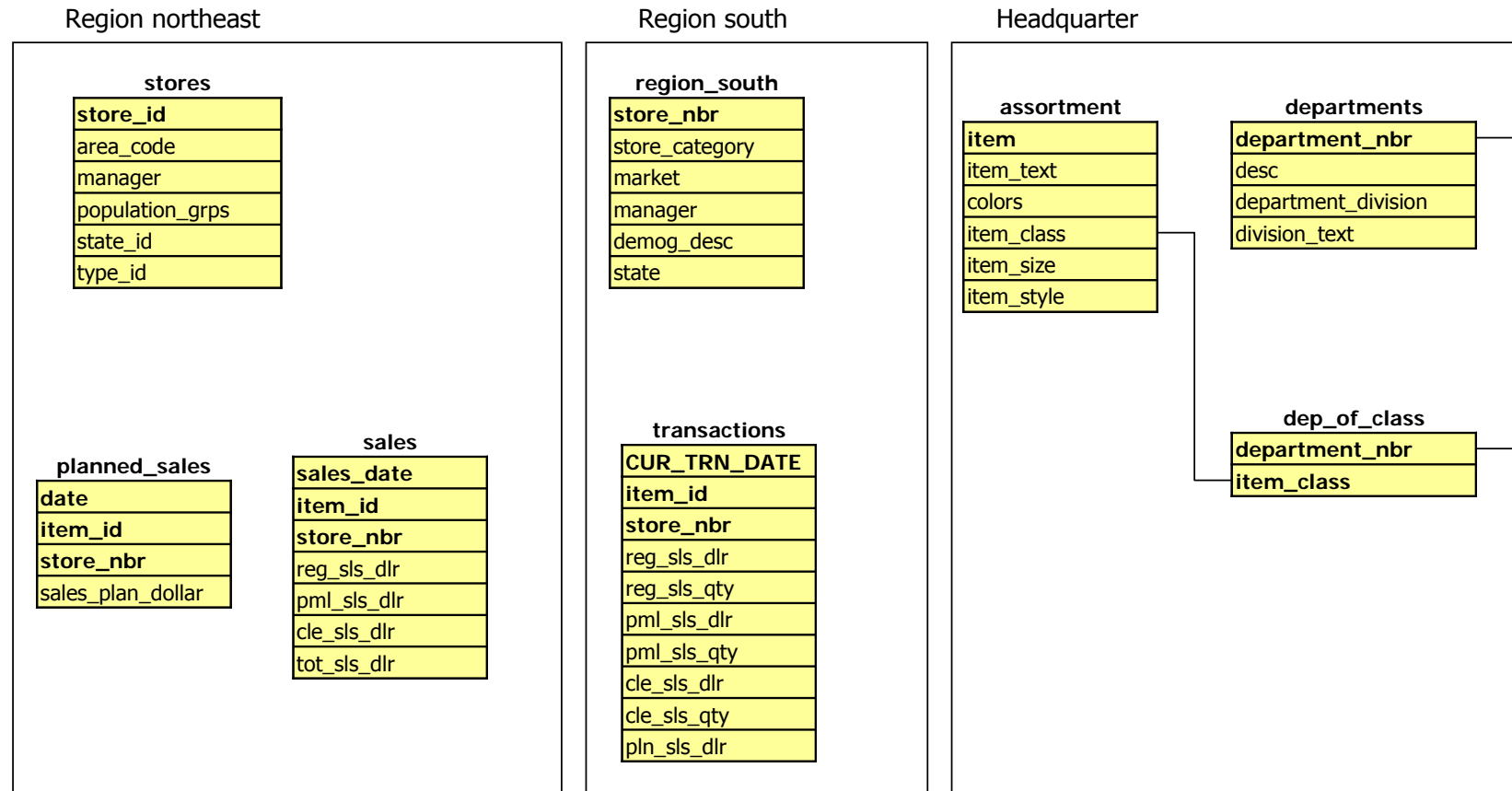
# Exercise 2

Data Integration and Monitoring

# Task 1: Intro ODPS

- In later exercises you will create and execute SQL queries. We use the ODPS system for this purpose. ODPS basically is a web interface to an IBM DB2 database. The necessary password for registration in ODPS (course **DWDM2015**) is announced during the exercise lesson. The document **ODPS\_Intro\_v04.pdf** gives a short introduction to ODPS which is available here:  
**<https://odps.informatik.uni-stuttgart.de>**
- ODPS also includes a tutorial called **Simple Queries in SQL** covering the main types of SQL queries you should be familiar with for this and the following exercises. If you are not sure about your SQL skills, it is recommended that you go through this tutorial as soon as possible.

# Task 2



The clothing company trendy4all operates branch offices that are each assigned to one of the regions "South" and "North East". For each of the regions, data is managed in a separate database covering each store and the sales that take place there. In addition, in the headquarter and also in each region the complete data on all items offered by trendy4all is available.

## Task 2

Table	Column	Description
stores	store_id	For each store that is a unique number. A textual description of the stores is not available in the database.
	area_code	It indicates, which region the store belongs to, e.g. 1 for "Mid-Atlantic", 2 for "New England"...
	manager	Name of the responsible manager
	population_grps	Population groups that typically go shopping in this store. An entry consists of a list of numbers separated by a comma. They indicate a group, e.g. 3 for "Latino", 7 for "Hispanic"...
	state_id	Number of the state where the store is located
	type_id	Type of store, e.g. 1 for "Super Malls", 3 for "Retail Only".
planned_sales	date	date
	item_id	Product number
	store_nbr	Store number
	sales_plan_dollar	Planned turnover for the indicated day, product and store. Turnover is in dollars.
sales	sales_date	date
	item_id	Product number
	store_nbr	Store number
	reg_sls_dlr	Turnover in dollars of regular goods
	pml_sls_dlr	Turnover in dollars of regular goods that are advertised
	cle_sls_dlr	Turnover in dollars of goods in closing sale
	total_sls_dlr	Total turnover



## Task 2

Table	Column	Description
region_south	store_nbr	Unique number of a store within region „south“
	store_category	Category of a store, e.g. „Wholesale Only“, „Retail Only“ or „Super Malls“.
	market	Part of the region where the store is located, e.g. „Deep South“
	manager	Name of responsible manager
	demo_desc	Population groups that typically go shopping in this store. An entry consists of a list of numbers separated by a comma. They indicate a group, e.g. 3 for „Latino“, 7 for „Hispanic“...
	state	Two letters for the state, where the store is located
transactions	cur_trn_date	Date of transaction
	item_id	Product number
	store_nbr	Store number
	reg_sls_dlr	Turnover and quantity of regular goods
	reg_sls_qty	
	pml_sls_dlr	Turnover and quantity of regular goods that are advertised
	pml_sls_qty	
	cle_sls_dlr	Turnover and quantity of goods in closing sale
	cle_sls_qty	
	pln_sls_dlr	Planned total turnover

## Task 2

Table	Column	Description
assortment	item	Product number
	item_text	Product description
	colors	List of colors available for the product
	item_class	Assigned product group
	item_size	Indication of size of the product (from 1= small to 5=X-Large)
	item_style	Describes textually the season in which the product is used (Winter, Spring, Summer).
departments	department_nbr	Department number
	desc	Description of the department
	department_division	Number of the department division
	division_text	Description of the department division, e.g. „Casual Clothing“

## Task 3

- a) Explore how stores are identified in each region.
- b) Compare how states and various kinds of stores are represented in the databases.
- c) Which regions are relevant for trendy4all?
- d) Analyze available population groups.
- e) Compare the sales information.

Explore the data with respect to the mentioned aspects. Think about the possible issues that may arise when you are asked to integrate the data of trendy4all in one database.

## Task 3a)

- `SELECT DISTINCT store_id FROM stores`
- `SELECT DISTINCT store_nbr FROM region_south`
- Both queries deliver IDs from 1 to 5.  
Hence, there are no globally unique IDs for stores.
- Looking at the schema reveals that
  - `store_id` is defined as `INTEGER`
  - `store_nbr` is defined as `BIGINT`
- Conclusion:
  - **Keys** available in `Store_id` and `store_nbr` have to be consolidated when data is integrated.
  - **Data types** have to be aligned.

STORE_ID
1
2
3
4
5

STORE_NBR
1
2
3
4
5

## Task 3b)

- States
  - In table stores, there is only one state with id 1. In region south, there is only one state 'CT'.
  - Obviously, states are **encoded** in different ways. Hence, the encoding has to be aligned.
- Kind of stores
  - Attributes store\_category and type\_id seem to refer to various kinds of stores.
  - **Semantics** of these attributes needs to be checked.
  - These kinds of stores are also **encoded** in different ways.
  - Some kind of mapping is necessary here as well.

STORE_CATEGORY
Retail Only
Strip Malls
Super Malls
Wholesale Only

TYPE_ID
1
2
3
4

## Task 3c)

- `SELECT DISTINCT area_code FROM stores`
- In northeast two areas (or regions as they are called in the data explanation) are mentioned (1 for "Mid-Atlantic", 2 for "New England")
- The corresponding information is **missing** for region south as no special areas are available. Markets are described as parts of regions.
- Hence, 3 regions seem to be relevant:
  - Mid-Atlantic
  - New England
  - South

AREA_CODE
1
2

## Task 3d)

- SELECT population\_grps FROM stores
- SELECT demog\_desc FROM region\_south

POPULATION_GRP	DEMO_DESC
2, 3, 4,	3, 4,
1, 3, 4, 5, 6, 7,	1,
3, 4,	1,
1, 3, 4, 7,	1, 2, 3,
1, 2, 3, 4, 5, 6, 7,	2,

- Both attributes seem to reflect the same information.
- **Encoding** needs to be checked and perhaps aligned.
- Combined information in these columns needs to be separated.

## Task 3e)

- Sales attributes in sales, planned\_sales and transactions are based on the same data type.
- The **unit** of turnover seems to be Dollar – at least this is indicated by the attribute name (needs to be verified)
- Turnover data is available on the same **granularity**: per day, per item and per store.
- Turnover is available in both regions.
- Quantity of goods is available in region south and **missing** for the other regions.
- Data integration
  - Focus on data available in both regions
  - Add data source that provide additional information, e.g., information on quantity of goods in our case



## Task 4

- Given the database schema of trendy4all as described in Task 1.2. We want to derive the following aggregated information for trendy4all:
  1. Provide an overview of stores from all regions.
  2. Aggregate turnover and quantity (regular, promotional, clearance, total) on the department and store level. We also need this aggregation per year and month.
- Discuss the difficulties that could arise when we try to provide this aggregated information. Provide the requested data using a single SQL query (if possible).

# Overview Stores

- stores

STORE_ID	AREA_CODE	MANAGER	POPULATION_GRP5	STATE_ID	TYPE_ID
4	1	Joe Manager	2, 3, 4,	1	4
3	2	Joe Manager	1, 3, 4, 5, 6, 7,	1	3
5	1	Joe Manager	3, 4,	1	3
2	2	Joe Manager	1, 3, 4, 7,	1	2
1	2	Joe Manager	1, 2, 3, 4, 5, 6, 7,	1	1

- region\_south

STORE_NBR	STORE_CATEGORY	MARKET	MANAGER	DEMO_DESC	STATE
4	Wholesale Only	Deep South	Jim Manager	3, 4,	CT
2	Retail Only	Carolinas	Jm Manager	1,	CT
3	Strip Malls	Carolinas	Jim Manager	1,	CT
5	Super Malls	Deep South	Jim Manager	1, 2, 3,	CT
1	Super Malls	Carolinas	Jim Manager	2,	CT

```

SELECT case when area_code=1 then 'Mid-Atlantic' when area_code=2 then 'New England' else
NULL end as region,
case when state_id=1 then 'NY' else NULL end as state,
store_id as id,
manager,
case when type_id=1 then 'Super Malls' when type_id=3 then 'Retail Only' else 'no category'
end as category,
population_grps as demographics
FROM stores

```

### UNION

```

SELECT 'South' as region,
state,
store_nbr as id,
manager,
store_category as category,
demo_desc as demographics
FROM region_south

```

REGION	STATE	ID	MANAGER	CATEGORY	DEMOGRAPHICS
South	CT	1	Jim Manager	Super Malls	2,
New England	NY	1	Joe Manager	Super Malls	1, 2, 3, 4, 5, 6, 7,
South	CT	2	Jm Manager	Retail Only	1,
New England	NY	2	Joe Manager	no category	1, 3, 4, 7,
New England	NY	3	Joe Manager	Retail Only	1, 3, 4, 5, 6, 7,
South	CT	3	Jim Manager	Strip Malls	1,
South	CT	4	Jim Manager	Wholesale Only	3, 4,
Mid-Atlantic	NY	4	Joe Manager	no category	2, 3, 4,
Mid-Atlantic	NY	5	Joe Manager	Retail Only	3, 4,
South	CT	5	Jim Manager	Super Malls	1, 2, 3,

```

SELECT      year, month, department, store_nbr,
            SUM(reg_sls_dlr) AS reg_sls_dlr,SUM(reg_sls_qty) AS reg_sls_qty,
            SUM(tot_sls_dlr) AS tot_sls_dlr,SUM(pln_sls_dlr) AS pln_sls_dlr

FROM (
SELECT      year(sales.sales_date)AS year,month(sales.sales_date) AS month,
            departments.desc AS department,sales.store_nbr AS store_nbr,
            reg_sls_dlr AS reg_sls_dlr, 0 AS reg_sls_qty,
            tot_sls_dlr AS tot_sls_dlr,
            sales_plan_dollar AS pln_sls_dlr
FROM        planned_sales,sales,assortment,dep_of_class,departments
WHERE       planned_sales.date = sales.sales_date
AND         planned_sales.item_id = sales.item_id
AND         planned_sales.store_nbr = sales.store_nbr
AND         planned_sales.item_id = assortment.item
AND         assortment.item_class = dep_of_class.item_class
AND         dep_of_class.item_class = departments.department_nbr
UNION ALL
SELECT      year(transactions.cur_trn_date) AS year,month(transactions.cur_trn_date) AS month,
            departments.desc AS department,transactions.store_nbr AS store_nbr,
            reg_sls_dlr AS reg_sls_dlr,reg_sls_qty AS reg_sls_qty,
            reg_sls_dlr+pml_sls_dlr+cle_sls_dlr AS tot_sls_dlr,
            pln_sls_dlr AS pln_sls_dlr
FROM        transactions,assortment,dep_of_class,departments
WHERE       transactions.item_id = assortment.item
AND         assortment.item_class = dep_of_class.item_class
AND         dep_of_class.item_class = departments.department_nbr
            ) AS temp
GROUP BY    year, month, department, store_nbr;

```

```

SELECT      year, month, department, store_nbr,
            SUM(reg_sls_dlr) AS reg_sls_dlr, SUM(reg_sls_qty) AS reg_sls_qty,
            SUM(tot_sls_dlr) AS tot_sls_dlr, SUM(pln_sls_dlr) AS pln_sls_dlr

FROM (
SELECT      year(sales.sales_date) AS year, month(sales.sales_date) AS month,
            departments.desc AS department, map_store_nbr1(sales.store_nbr) AS store_nbr,
            reg_sls_dlr AS reg_sls_dlr, 0 AS reg_sls_qty,
            tot_sls_dlr AS tot_sls_dlr,
            sales_plan_dollar AS pln_sls_dlr
FROM        planned_sales, sales, assortment, dep_of_class, departments
WHERE       planned_sales.date = sales.sales_date
AND         planned_sales.item_id = sales.item_id
AND         planned_sales.store_nbr = sales.store_nbr
AND         planned_sales.item_id = assortment.item
AND         assortment.item_class = dep_of_class.item_class
AND         dep_of_class.item_class = departments.department_nbr
UNION ALL
SELECT      year(transactions.cur_trn_date) AS year, month(transactions.cur_trn_date) AS month,
            departments.desc AS department, map_store_nbr2(transactions.store_nbr) AS store_nbr,
            reg_sls_dlr AS reg_sls_dlr, reg_sls_qty AS reg_sls_qty,
            reg_sls_dlr+pml_sls_dlr+cle_sls_dlr AS tot_sls_dlr,
            pln_sls_dlr AS pln_sls_dlr
FROM        transactions, assortment, dep_of_class, departments
WHERE       transactions.item_id = assortment.item
AND         assortment.item_class = dep_of_class.item_class
AND         dep_of_class.item_class = departments.department_nbr ) AS temp
GROUP BY    year, month, department, store_nbr;

```

```
SELECT      year(sales.sales_date) AS year, month(sales.sales_date) AS month,
            departments.desc AS department, sales.store_nbr AS store_nbr,
            reg_sls_dlr AS reg_sls_dlr, 0 AS reg_sls_qty,
            tot_sls_dlr AS tot_sls_dlr,
            sales_plan_dollar AS pln_sls_dlr
FROM        planned_sales, sales, assortment, dep_of_class, departments
WHERE       planned_sales.date = sales.sales_date
AND         planned_sales.item_id = sales.item_id
AND         planned_sales.store_nbr = sales.store_nbr
AND         planned_sales.item_id = assortment.item
AND         assortment.item_class = dep_of_class.item_class
AND         dep_of_class.item_class = departments.department_nbr
GROUP BY   year(sales.sales_date), month(sales.sales_date), departments.desc, sales.store_nbr
```

UNION ALL

```
SELECT      year(transactions.cur_trn_date) AS year, month(transactions.cur_trn_date) AS month,
            departments.desc AS department, transactions.store_nbr AS store_nbr,
            reg_sls_dlr AS reg_sls_dlr, reg_sls_qty AS reg_sls_qty,
            reg_sls_dlr+pml_sls_dlr+cle_sls_dlr AS tot_sls_dlr,
            pln_sls_dlr AS pln_sls_dlr
FROM        transactions, assortment, dep_of_class, departments
WHERE       transactions.item_id = assortment.item
AND         assortment.item_class = dep_of_class.item_class
AND         dep_of_class.item_class = departments.department_nbr
GROUP BY   year(transactions.cur_trn_date), month(transactions.cur_trn_date), departments.desc,
            transactions.store_nbr;
```

```
SELECT      year(sales.sales_date) AS year, month(sales.sales_date) AS month,
            departments.desc AS department, map_store_nbr1(sales.store_nbr) AS store_nbr,
            reg_sls_dlr AS reg_sls_dlr, 0 AS reg_sls_qty,
            tot_sls_dlr AS tot_sls_dlr,
            sales_plan_dollar AS pln_sls_dlr
FROM        planned_sales, sales, assortment, dep_of_class, departments
WHERE       planned_sales.date = sales.sales_date
AND         planned_sales.item_id = sales.item_id
AND         planned_sales.store_nbr = sales.store_nbr
AND         planned_sales.item_id = assortment.item
AND         assortment.item_class = dep_of_class.item_class
AND         dep_of_class.item_class = departments.department_nbr
GROUP BY year(sales.sales_date), month(sales.sales_date), departments.desc, sales.store_nbr
```

UNION ALL

```
SELECT      year(transactions.cur_trn_date) AS year, month(transactions.cur_trn_date) AS month,
            departments.desc AS department, map_store_nbr2(transactions.store_nbr) AS store_nbr,
            reg_sls_dlr AS reg_sls_dlr, reg_sls_qty AS reg_sls_qty,
            reg_sls_dlr+pml_sls_dlr+cle_sls_dlr AS tot_sls_dlr,
            pln_sls_dlr AS pln_sls_dlr
FROM        transactions, assortment, dep_of_class, departments
WHERE       transactions.item_id = assortment.item
AND         assortment.item_class = dep_of_class.item_class
AND         dep_of_class.item_class = departments.department_nbr
GROUP BY year(transactions.cur_trn_date), month(transactions.cur_trn_date), departments.desc,
transactions.store_nbr;
```

## Task 5: Monitoring

- Given the database schema of data sources of trendy4all as described in Task 2.2. To find the data the ETL process should move into the data warehouse, the monitoring component has to identify data changes in the source systems.
  1. Discuss under the following assumptions how monitoring could be achieved for trendy4all. Consider each set of assumptions separately.
    - a. All source systems use a centralized RDBMS for data management.
    - b. The source systems use a distributed RDBMS for data management.
    - c. All source systems store the data in a set of files in the file system.
    - d. Some of the source systems use files for data management, others use a centralized RDBMS.



# Monitoring

- Goal: Discover changes in data source incrementally
- Approaches:

	Based on ...	Changes identified by ...
Trigger	triggers defined in source DBMS	trigger writes a copy of changed data to files
Replica	replication support of source DBMS	replication provides changed rows in a separate table
Timestamp	timestamp assigned to each row	use timestamp to identify changes (supported by temporal DBMS)
Log	log of source DBMS	read log
Snapshot	periodic snapshot of data source	compare snapshots

# Monitoring

- **Assumption a:** All source systems use a centralized RDBMS for data management.
  - Trigger (if provided by RDBMS)
  - Log
  - Timestamp (schema of the source system has to be changed)
  - Snapshot (always possible, but better options provided by RDBMS)



# Monitoring

- **Assumption b:** The source systems use a distributed RDBMS for data management.
  - Same as for the centralized RDBMS
  - Replica mechanisms might be an option as well

# Monitoring

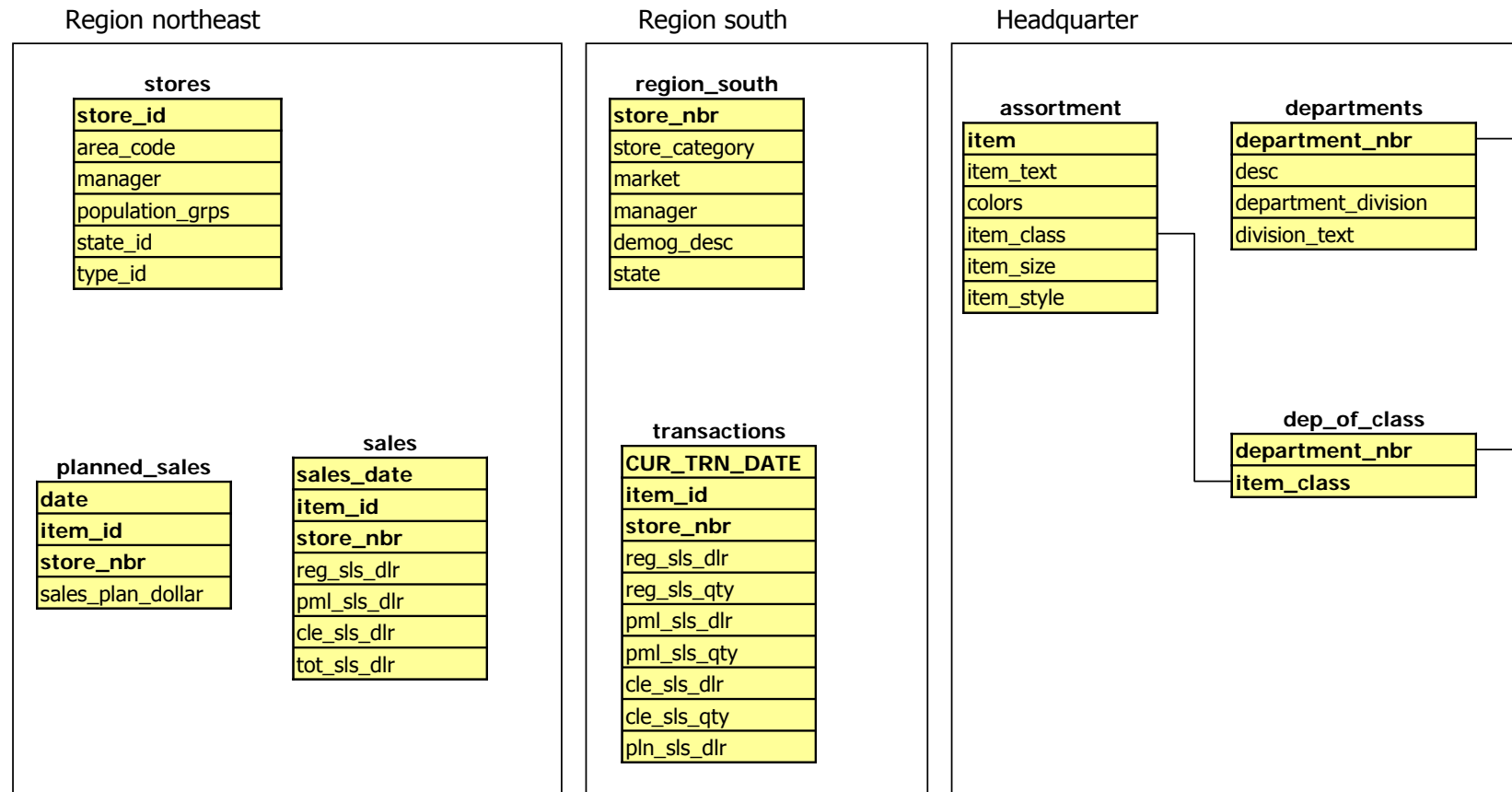
- **Assumption c:** All source systems store the data in a set of files in the file system.
  - Snapshot most likely the only option

# Monitoring

- **Assumption d:** Some of the source systems use files for data management, others use a centralized RDBMS.
  - No need to define an overall monitoring strategy
  - Decide on method to be used individually for each data source

## Task 5: Monitoring

- Given the database schema of data sources of trendy4all as described in Task 2.2. To find the data the ETL process should move into the data warehouse, the monitoring component has to identify data changes in the source systems.
- 2. We assume that the monitoring of these source systems is realized by triggers. Create the triggers that are necessary in the region “northeast” to log the changes of the source data in the following tables:
  - **Table new\_planned\_sales:** This table stores all new entries for the table planned\_sales. An additional column should store when data was added to table new\_planned\_sales.
  - **Table new\_sales:** This table stores all new entries of the table sales. Here, the time of the entry should also be stored.
  - **Table upd\_stores:** This table stores all changes in table stores. Both the type of change (insert/update/delete) and the time of change should be stored in additional columns.



The clothing company trendy4all operates branch offices that are each assigned to one of the regions "South" and "North East". For each of the regions, data is managed in a separate database covering each store and the sales that take place there. In addition, in the headquarter and also in each region the complete data on all items offered by trendy4all is available.

# Trigger 1



```
CREATE TRIGGER new_plan_sls
  AFTER INSERT ON planned_sales
  REFERENCING NEW AS new_row
  FOR EACH ROW
  INSERT INTO new_planned_sales
  VALUES(new_row.date,new_row.item_id,new_row.store_nbr,new
    w_row.sales_plan_dollar,CURRENT TIMESTAMP);
```



## Trigger 2

```
CREATE TRIGGER new_sls
  AFTER INSERT ON sales
  REFERENCING NEW AS new_row
  FOR EACH ROW
  INSERT INTO new_sales
  VALUES(new_row.sales_date,new_row.item_id,new_row.store_
    nbr,new_row.reg_sls_dlr,new_row.pml_sls_dlr,new_row.cl
    e_sls_dlr,new_row.tot_sls_dlr,CURRENT_TIMESTAMP);
```

## Trigger 3

```
CREATE TRIGGER new_strs
  AFTER INSERT ON stores
  REFERENCING NEW AS new_row
  FOR EACH ROW
  INSERT INTO upd_stores
  VALUES(new_row.store_id,new_row.area_code,new_row.manage
    r,new_row.population_grps,new_row.state_id,new_row.type_id,'I',CURRENT_TIMESTAMP);
```

# Trigger 4

```
CREATE TRIGGER del_strs
  AFTER DELETE ON stores
  REFERENCING OLD AS old_row
  FOR EACH ROW
  INSERT INTO upd_stores
  VALUES(old_row.store_id,old_row.area_code,old_row.manage
    r,old_row.population_grps,old_row.state_id,old_row.type
    e_id,'D',CURRENT_TIMESTAMP);
```

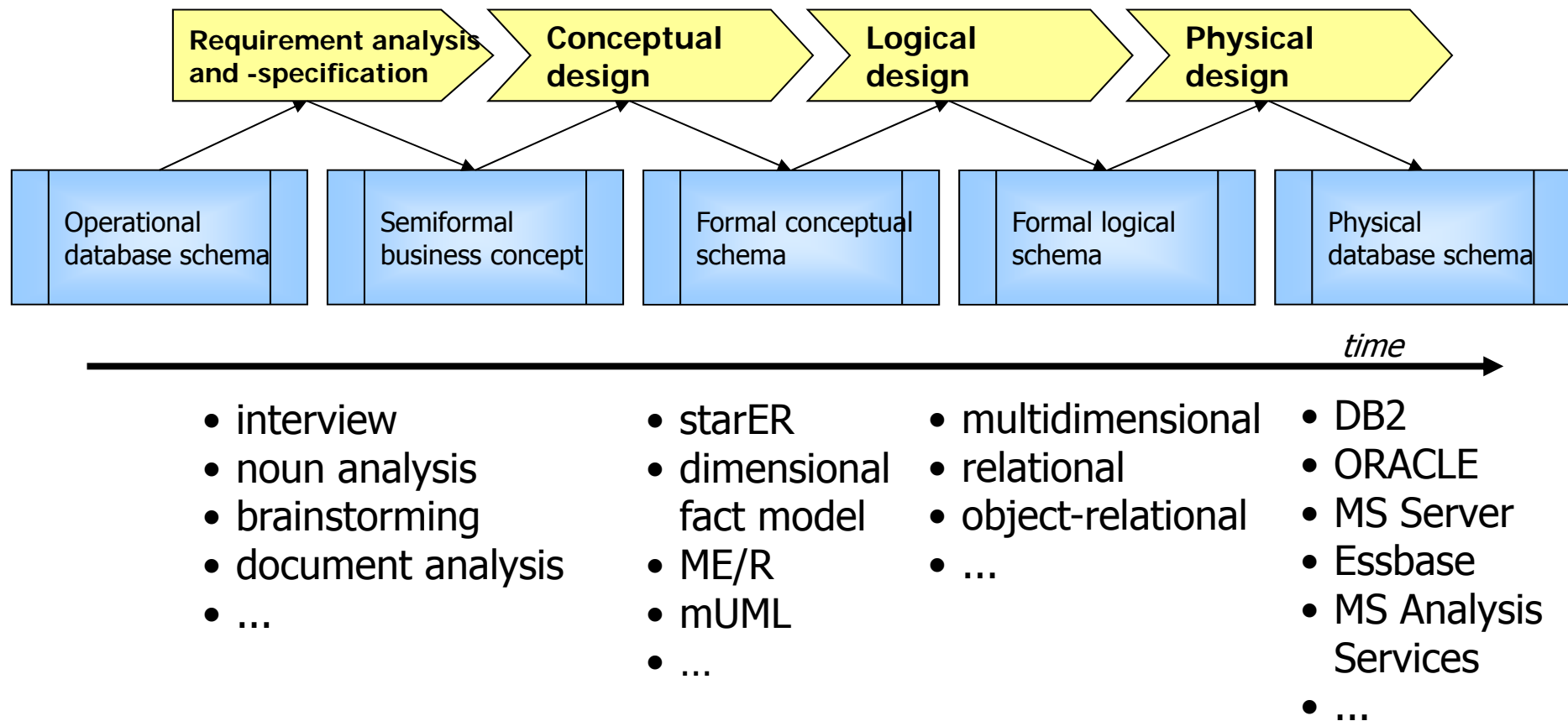
# Trigger 5

```
CREATE TRIGGER upd_strs
  AFTER UPDATE ON stores
  REFERENCING OLD AS old_row NEW AS new_row
  FOR EACH ROW
  INSERT INTO upd_stores
  VALUES(old_row.store_id,old_row.area_code,old_row.manager,
    old_row.population_grps,old_row.state_id,old_row.type_id, 'O', CURRENT_TIMESTAMP),
  (new_row.store_id,new_row.area_code,new_row.manager,new_row.population_grps,
    new_row.state_id,new_row.type_id, 'U', CURRENT_TIMESTAMP);
```

# Exercise 3

## Conceptual Data Warehouse Design

# Data Warehouse Design Process



# Task 1

- In this exercise, we cover the conceptual design of a data warehouse using the approaches introduced in the lecture (Dimensional Fact Model, UML Profile).
- The data warehouse should cover information related to the maintenance work on airplanes. Here is some information on the available data:
  - Maintenance is usually summarized in so-called **check types** (A, B, C, D). They are composed of many **work steps**. However, there are also independent maintenance operations that cannot be assigned to a check type.
  - For each work step, the **mechanics** that executed it are registered. In each work step, several mechanics from different **teams** may participate. They register timestamps for the **start** and the **end** of the work step. In the end, every work step is marked with a **state**, which describes, e.g., whether the operation had to be cancelled before completion. Mechanics are in particular characterized by their **name** and their **education**.
  - Maintenance work on air planes of different **years of construction**, **types** and **configurations** is done in a **hangar** next to the airport. In such a hangar, there are typically a number of **maintenance halls**. Owner of these hangars are **airlines**, but possibly also independent companies. Sometimes the hangars are also operated by multiple owners. Airplanes of a certain airline may also be checked in hangars of other airlines.
  - The data warehouse should support the analysis of maintenance work for **days**, **weeks**, **months**, **quarters**, and **years**. Additionally, a **business year** (October - September) should be considered.

# Task 1

- Based on this data warehouse, users should be able to derive the following information:
  - What is the average time for check type A in a maintenance hangar?
  - How many mechanics have participated in check type D for airbus A320 of a specific airline? Provide the plane's configuration as additional descriptive information.
  - In which months is most maintenance work executed? Report this for a specific maintenance hangar during the last year.
  - How often had work step 078/015 of the check type D to be interrupted?
  - Mechanic M was involved in the maintenance of how many planes last month/year/business year? Provide also the education level of the mechanic.
- 1. Create the conceptual design using the Dimensional Fact Model (graphical representation). Give examples of aggregation statements.
- 2. Use the UML Profile for creating the conceptual schema.
- 3. How do the DFM version and the UML version differ?



## Task 2

- Which facts and which dimensions have to be covered?
- Discuss the granularity of the facts with respect to the different dimensions.

# Basic Elements of a Conceptual Model

## Fact data

- Mostly numeric data that is observed or measured
- Example: turnover/sales, number of pieces, ...

## Qualities

- Represent a state, a status or a mode
- Cannot be aggregated (in contrast to fact data)
- Example: shipping mode, status

## Attributes

- Describe dimension objects
- Mostly text-based descriptions
- Example: product descriptions, customer profiles, addresses

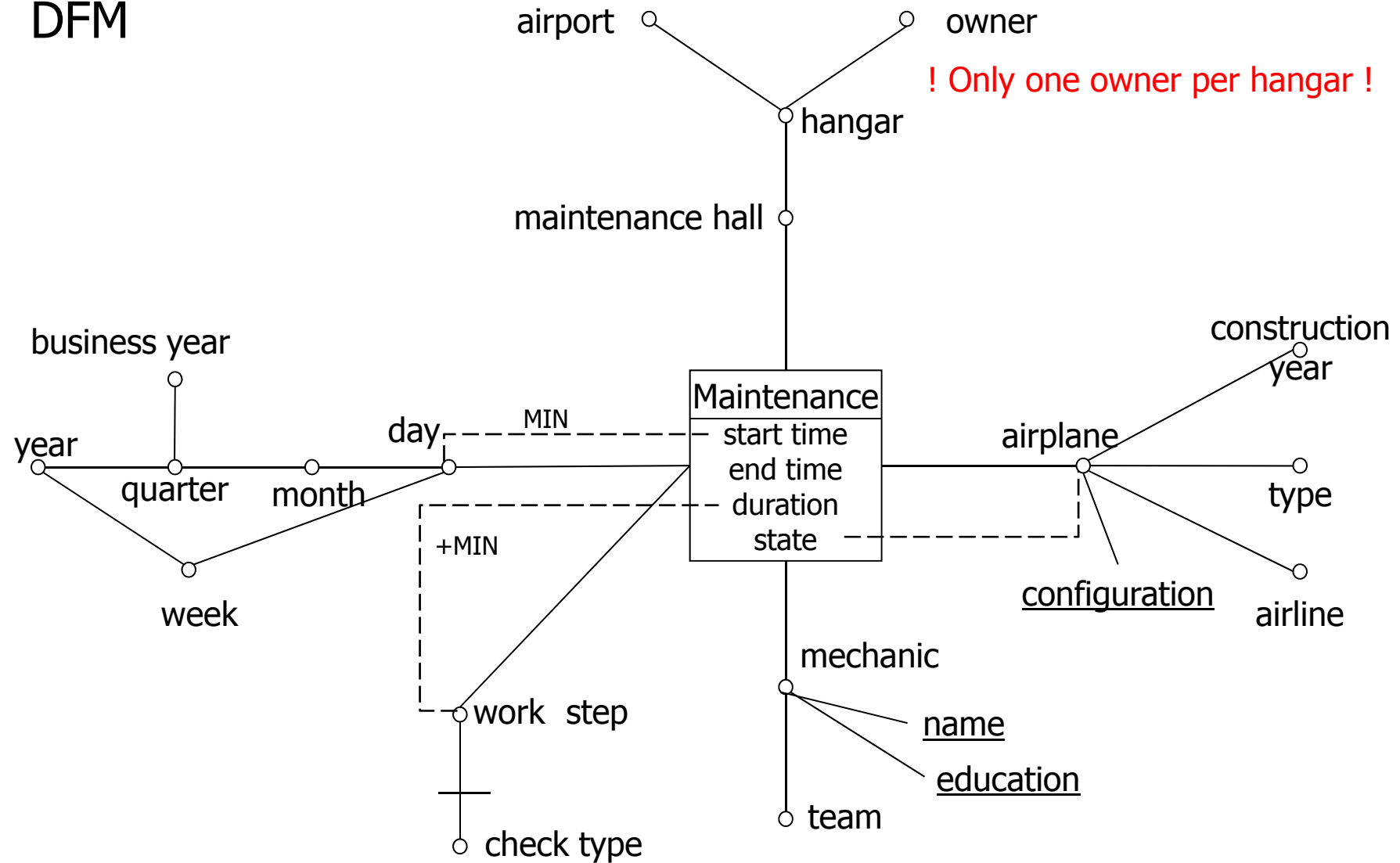
## Dimensions

- Strongly associated attributes
- Typically 5 to 20 attributes
- Showing mostly hierarchical relationships
- Example: product information, customer information, time references

## Task 3

- Create the conceptual design using the Dimensional Fact Model (graphical representation). Give examples of aggregation statements.

## DFM

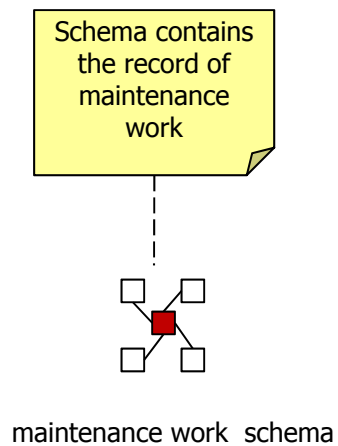


## Task 4

- Use the UML Profile for creating the conceptual schema. How do the DFM version and the UML version differ?

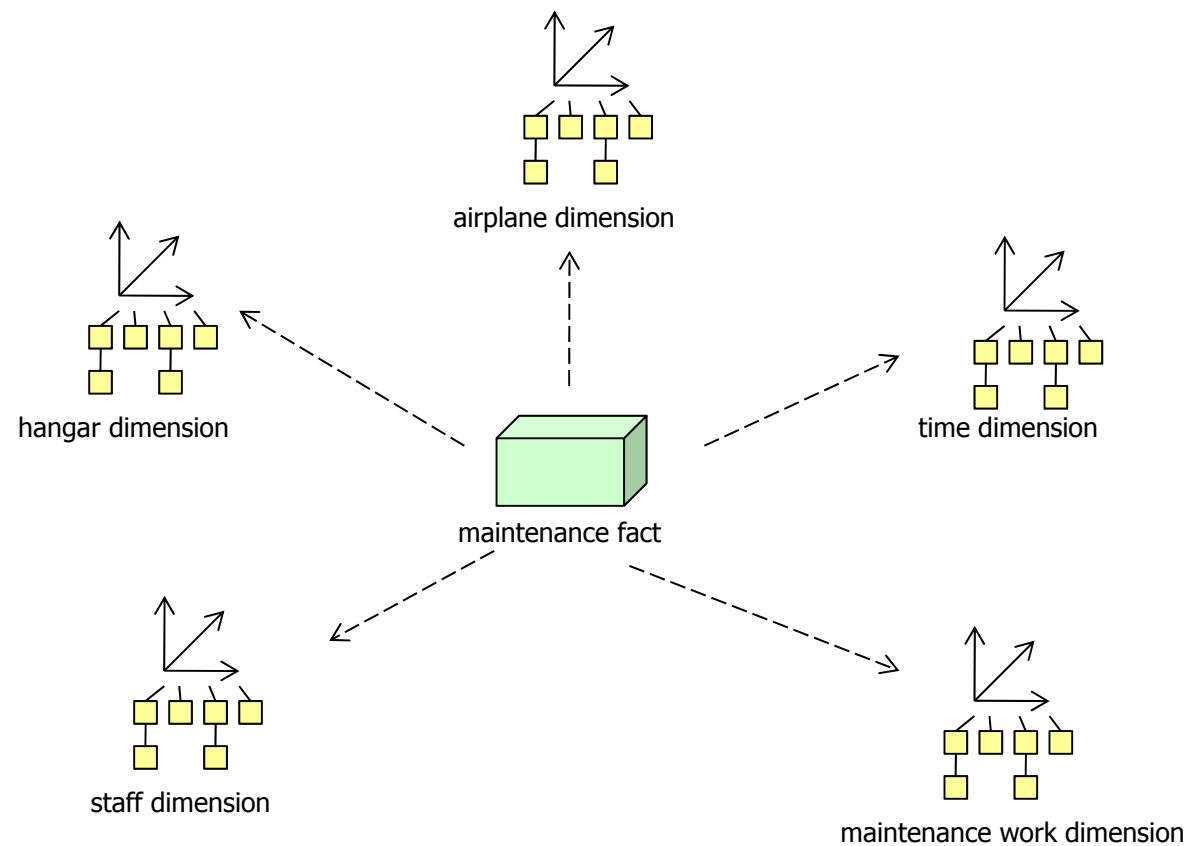
## Level 1

### Model Definition



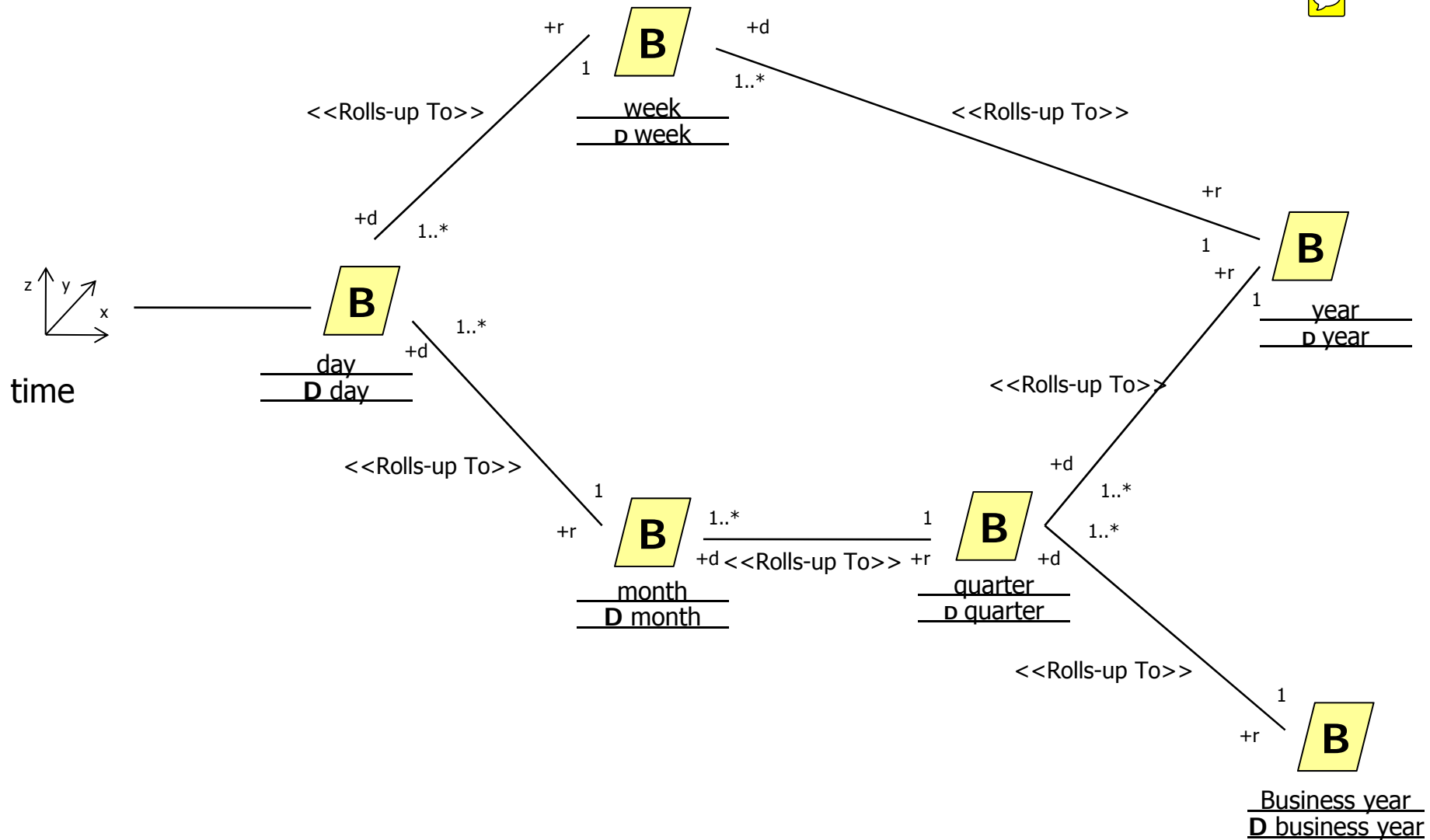
## Level 2

### Schema Definition



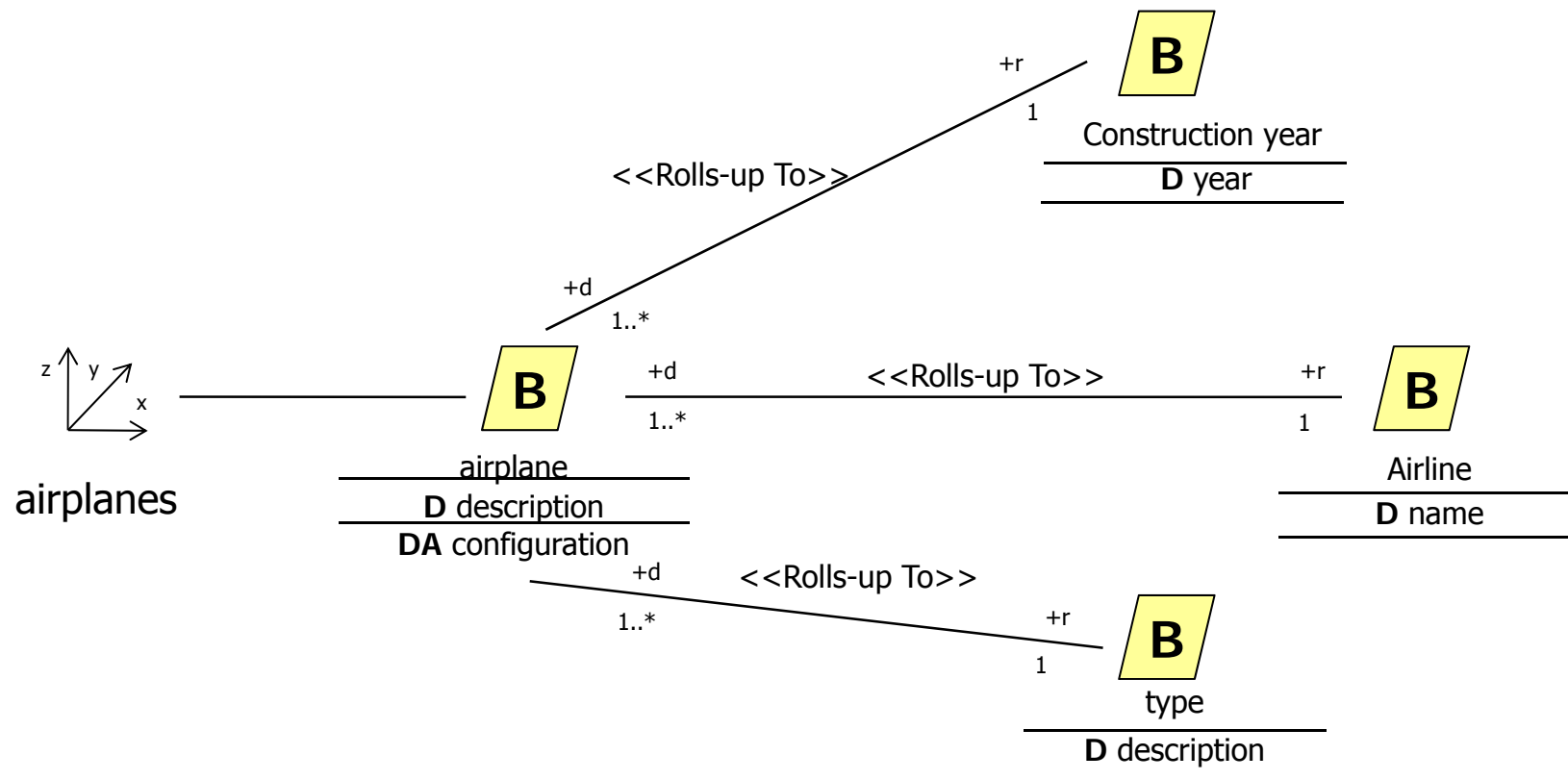
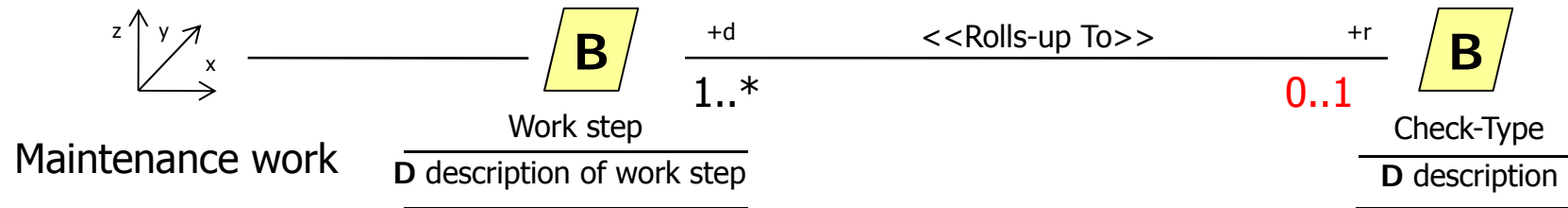


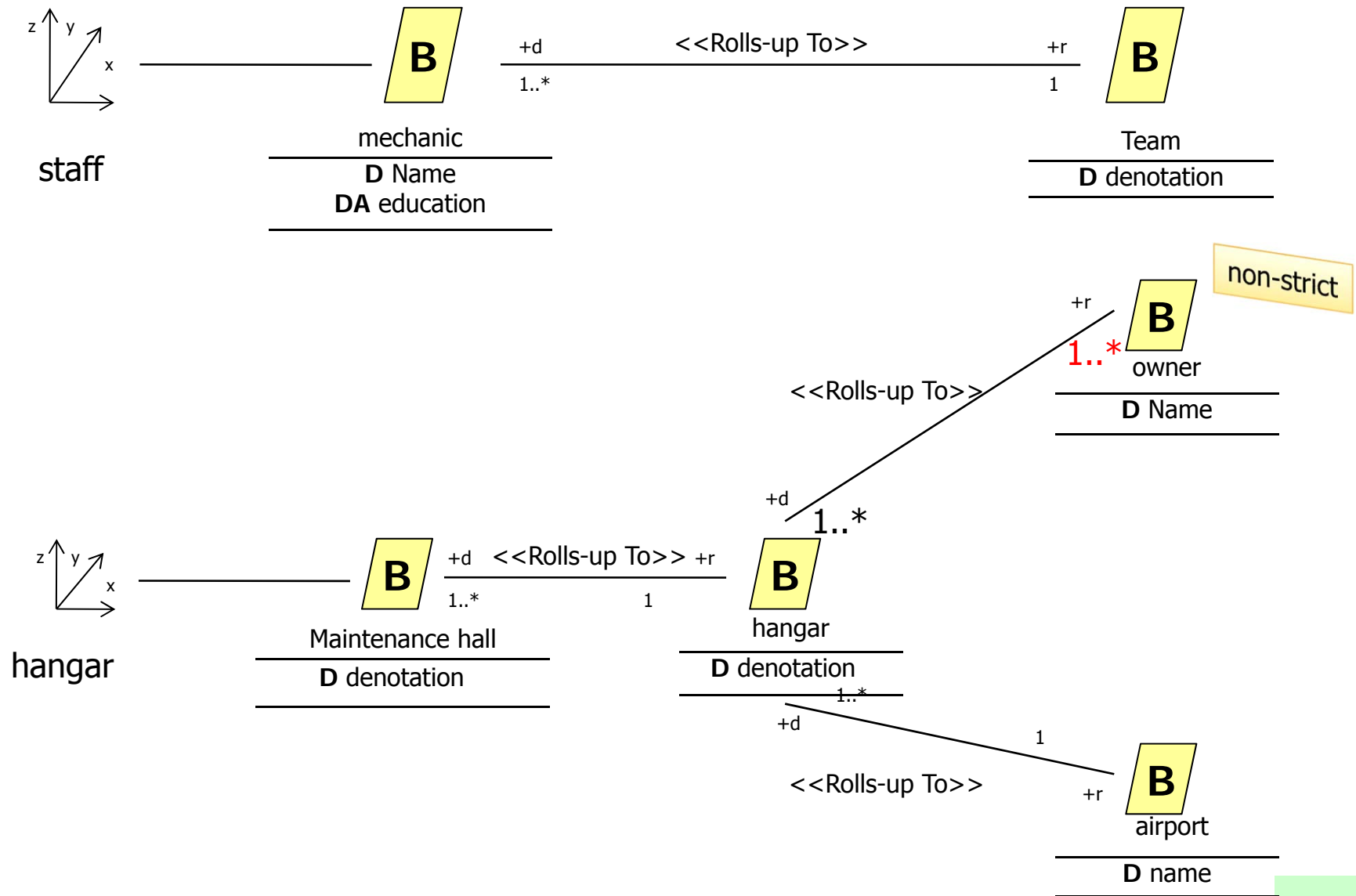
## Level 3: dimensions



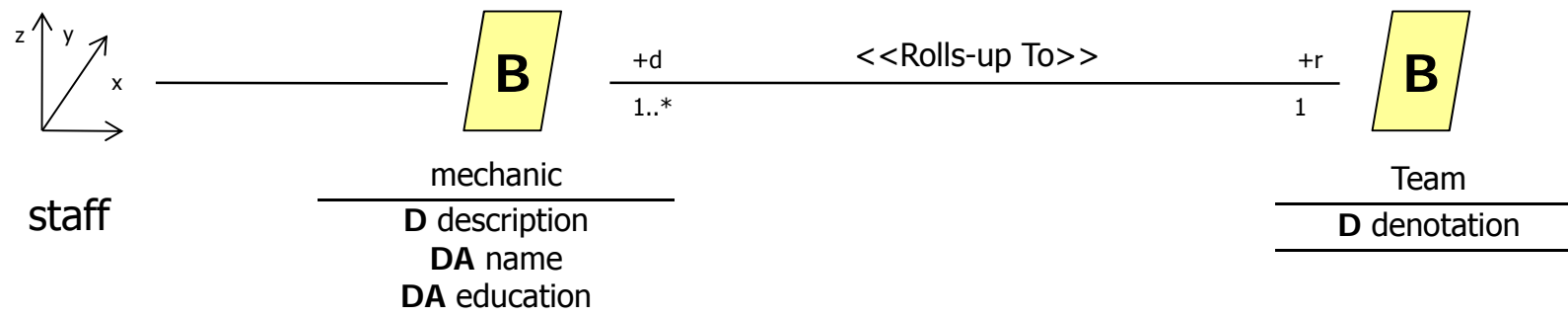
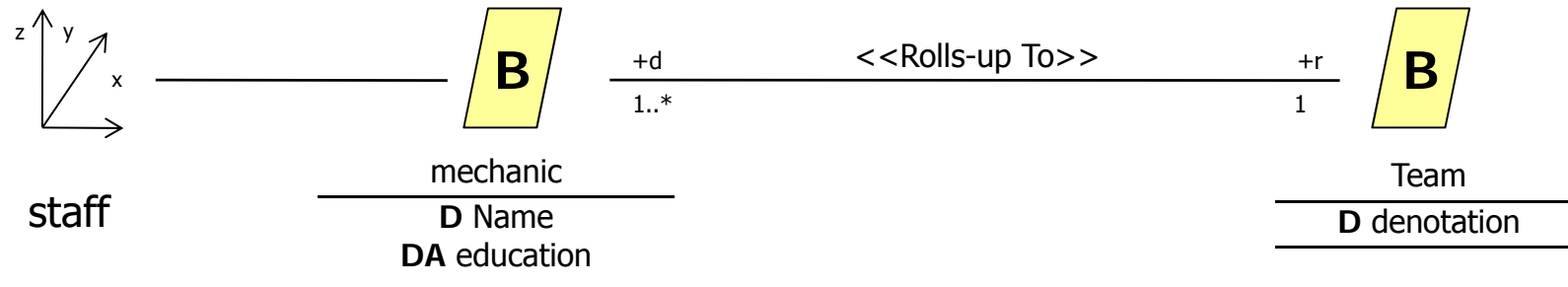


not complete!





# Alternative for staff dimension



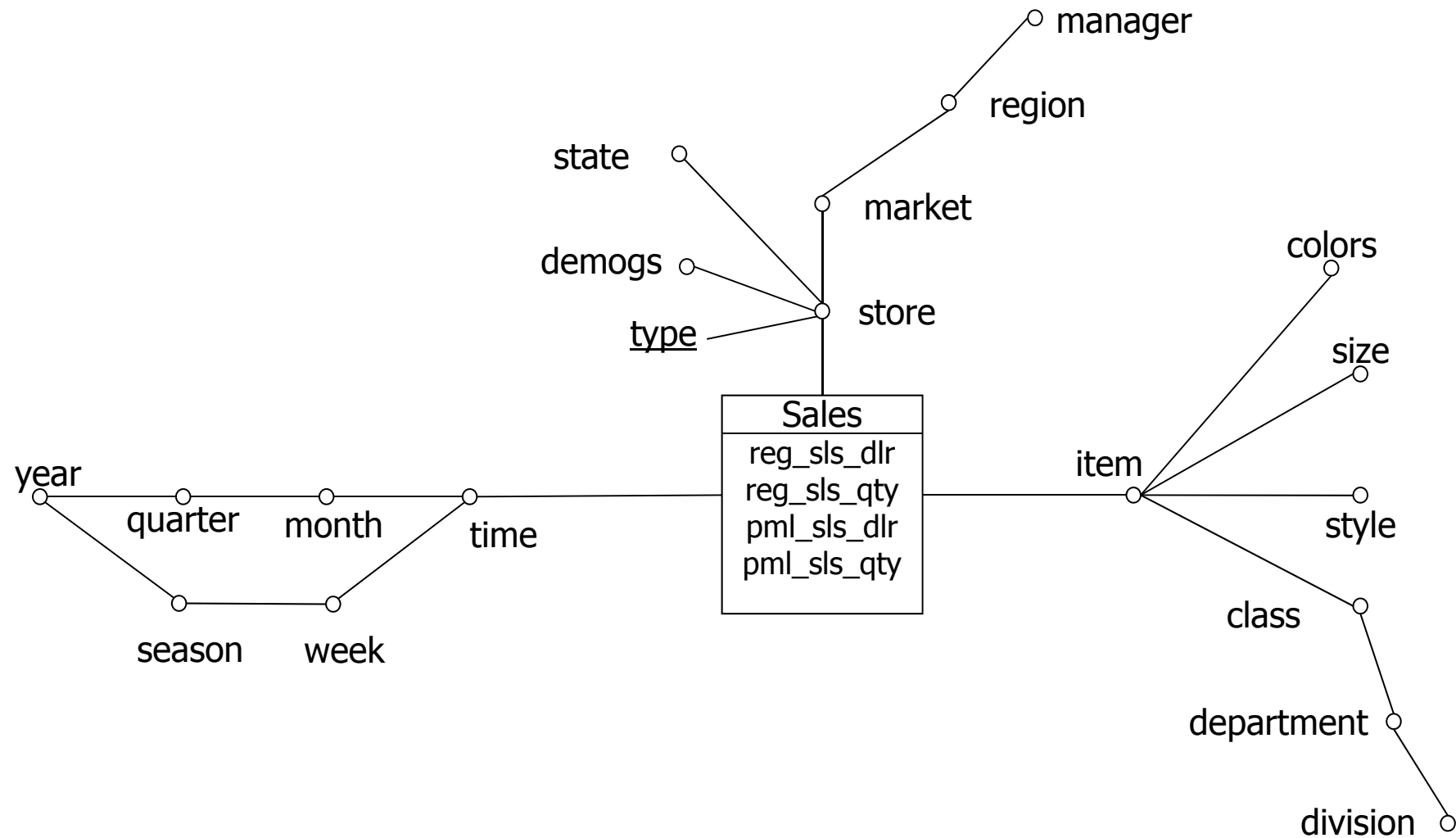
# Exercise 4

## Logical Data Warehouse Design

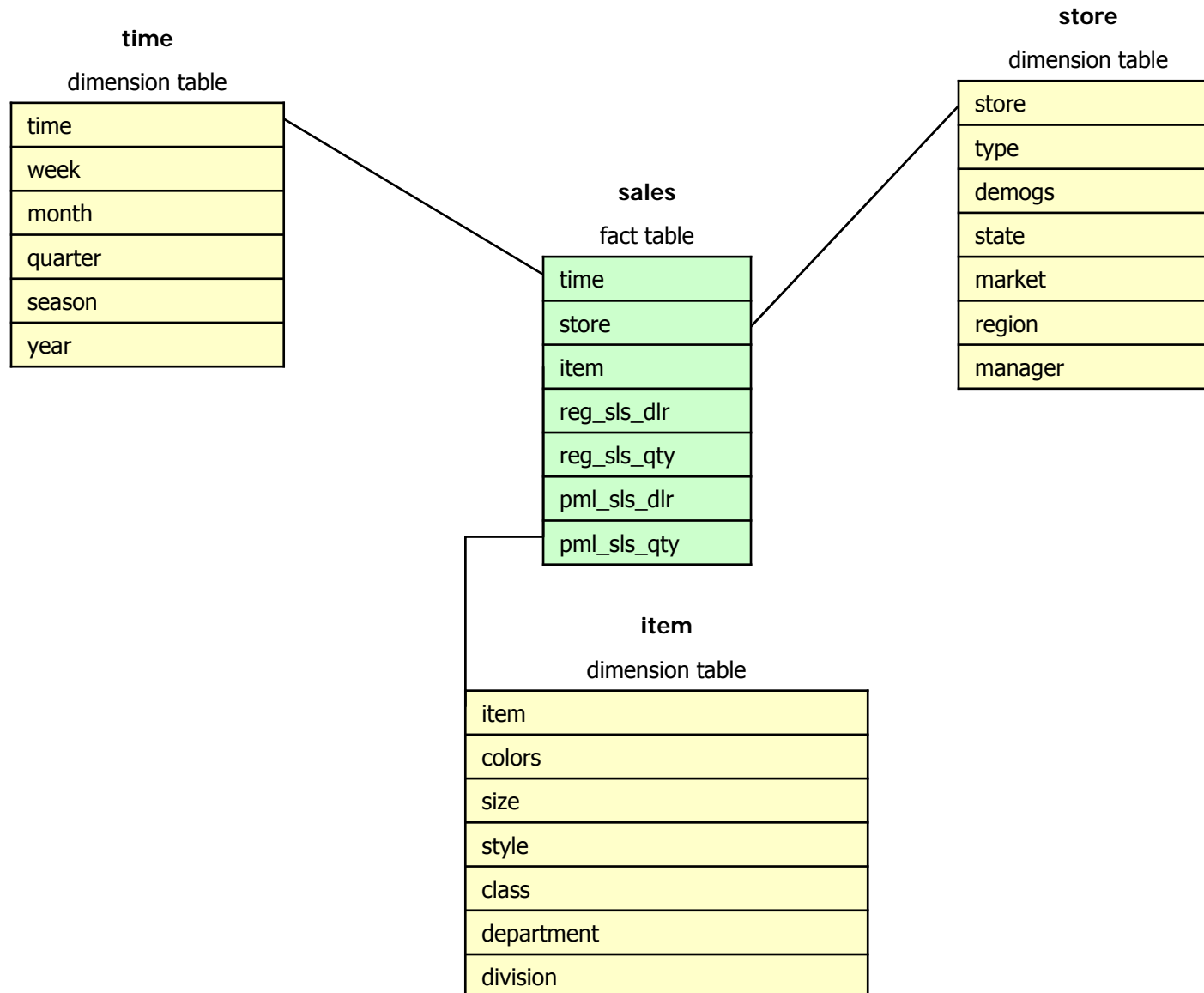
# Task 1

- The data trendy4all has to manage was already described in a previous exercise. The conceptual schema for trendy4all is shown in the following figure.
  1. Derive the logical schema from this conceptual schema using the star schema approach.
  2. Derive the logical schema from this conceptual schema using the snowflake schema approach.
  3. Name at least three aspects according to which the star schema and the snowflake schema differ. Compare both schema types with respect to these aspects.

# Conceptual Schema



# Star Schema

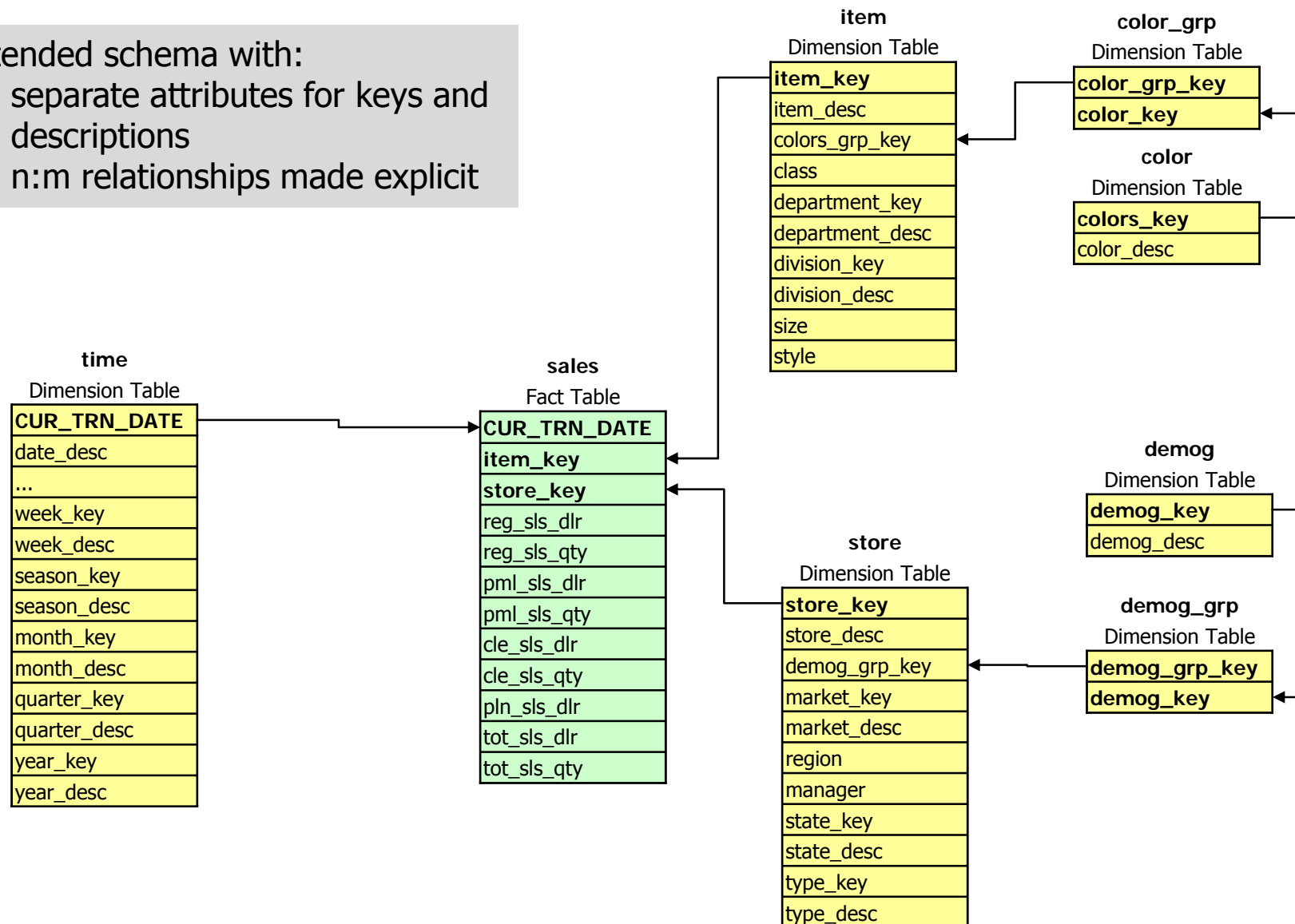


# Star Schema



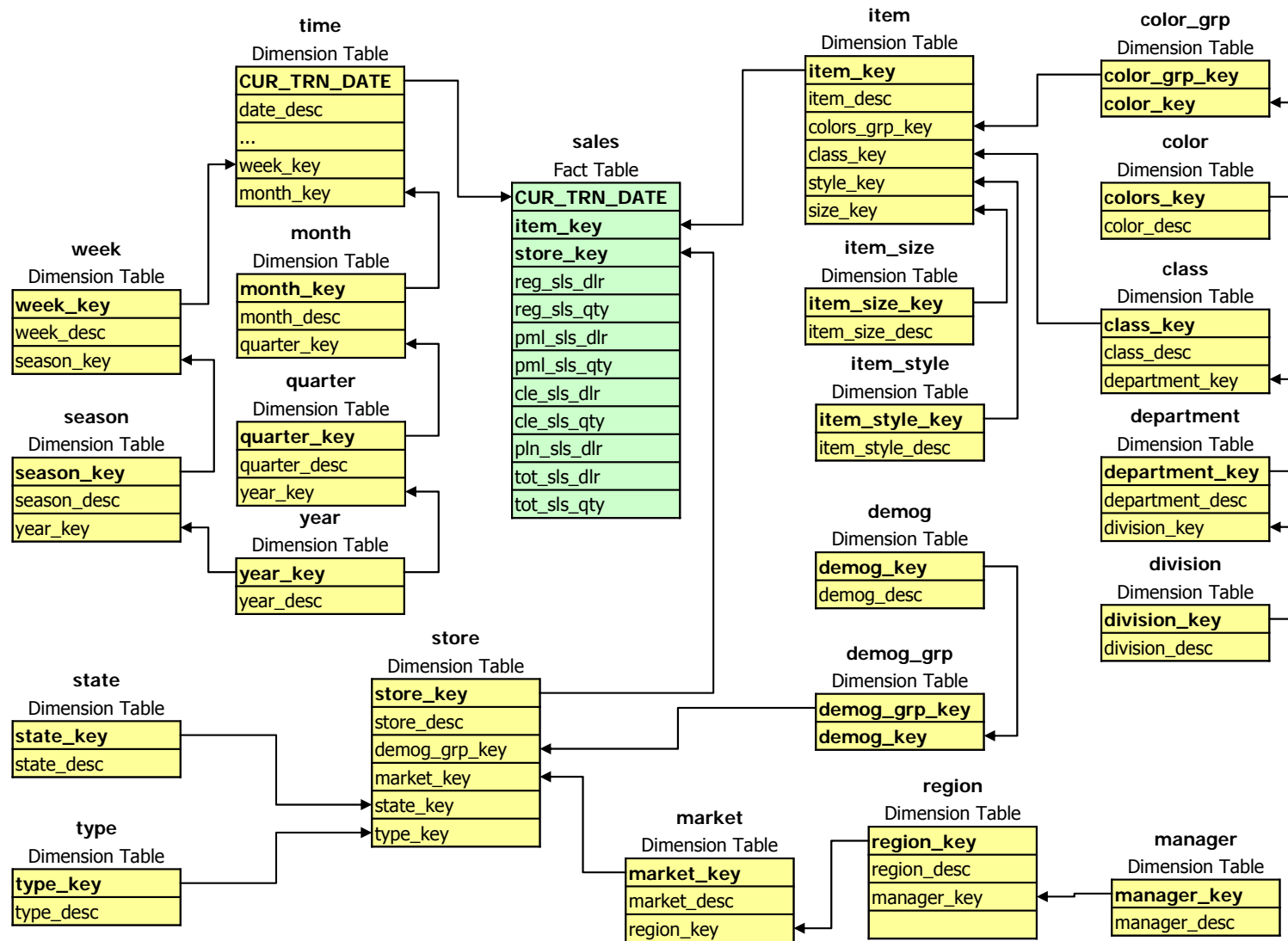
Extended schema with:

- separate attributes for keys and descriptions
- n:m relationships made explicit





# Snowflake Schema



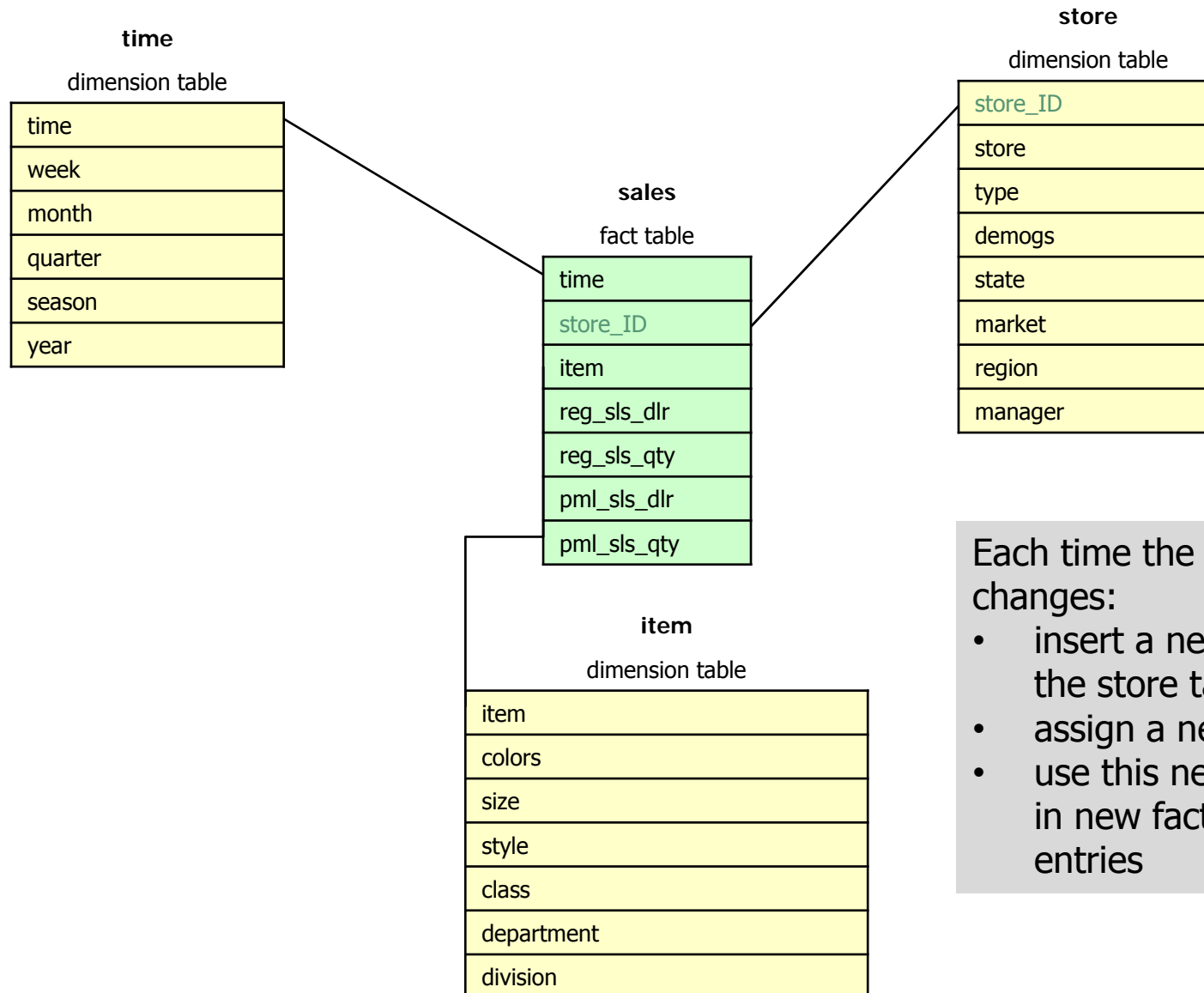
# Comparison

	Star Schema	Snowflake Schema	Informix Schema
Clearness	one table per dimension	multiple tables per dimension	multiple tables per dimension
Redundancy	in the dimension table	normalization takes care of	normalization takes care of
Data Volume	high(er)	low	low
Hierarchies	not represented	represented in dimension tables	only one hierarchy
Summarizability	-	-	-
Adding a Dimension	one additional table	several additional tables	several additional tables
Adding an Attribute	one attribute appended to dimension table	changes to several tables	changes to several tables
Performance (Queries)	max. one join per dimension	several joins among the dimension tables	several joins among the dimension tables

## Task 2

- Based on the star schema from task 4.1, consider the following aspects of extended dimension table design.
  1. The state of a store changes from time to time. We want to keep the complete history of state information over time. It should be possible to assign to each sales fact the store with the correct state information. There is no need to store when and for what reason the state information has changed. What is the appropriate design and usage of the store dimension table?
  2. All the detailed information stored for each item changes regularly. We want to keep the complete history. In particular, we want to track when and for what reason, the item data changes. What is the appropriate design and usage of the store dimension table?
  3. Comment on the following statements in one to three sentences:
    - There are three options to model slowly changing dimensions. All of them keep the complete history of changes.
    - The technique for modelling frequently changing large dimensions introduced in the lecture allows to keep all the details for each dimension attribute.
    - The technique for modelling frequently changing large dimensions introduced in the lecture results in frequent changes of the data volume of the dimension table.

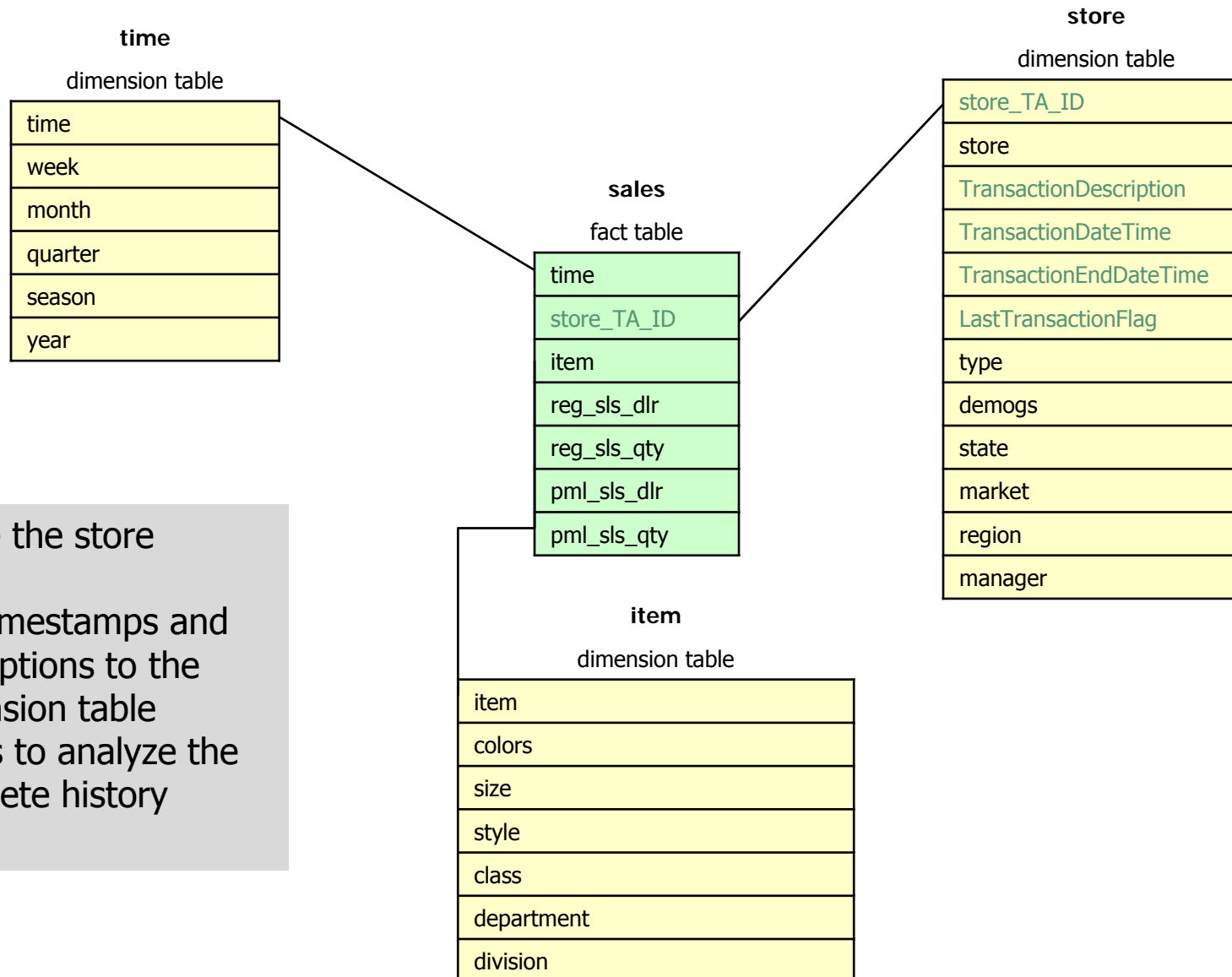
# Star Schema (for 1.)



Each time the store changes:

- insert a new row into the store table
- assign a new store\_ID
- use this new store\_ID in new fact table entries

# Star Schema (for 2.)



Each time the store changes:

- add timestamps and descriptions to the dimension table
- allows to analyze the complete history

# Large Dimensions with Frequent Changes

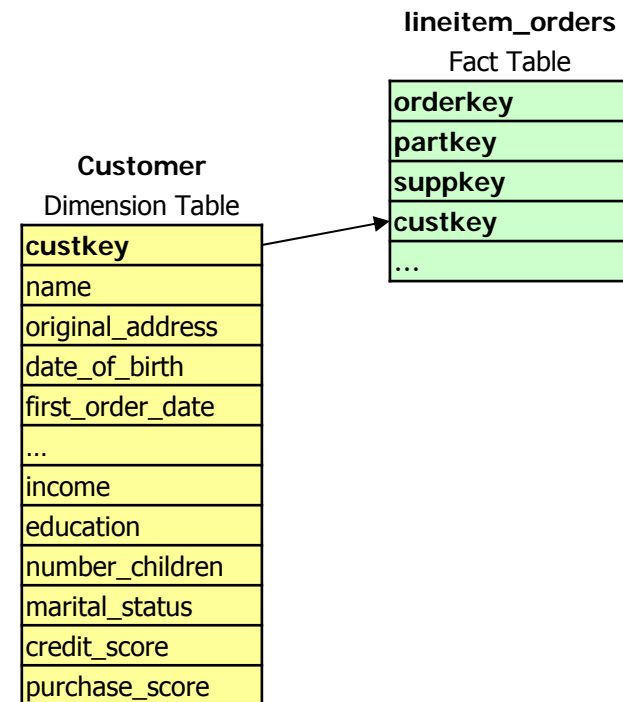
- **Example:**

- For an insurance company the customer dimension might hold millions of tuples; large amounts of demographic data are kept for each customer.
- Customer data change constantly and these changes have to be propagated into the warehouse.

Customer Dimension Table	
<b>custkey</b>	
name	
original_address	
date_of_birth	
first_order_date	
...	
income	
education	
number_children	
marital_status	
credit_score	
purchase_score	

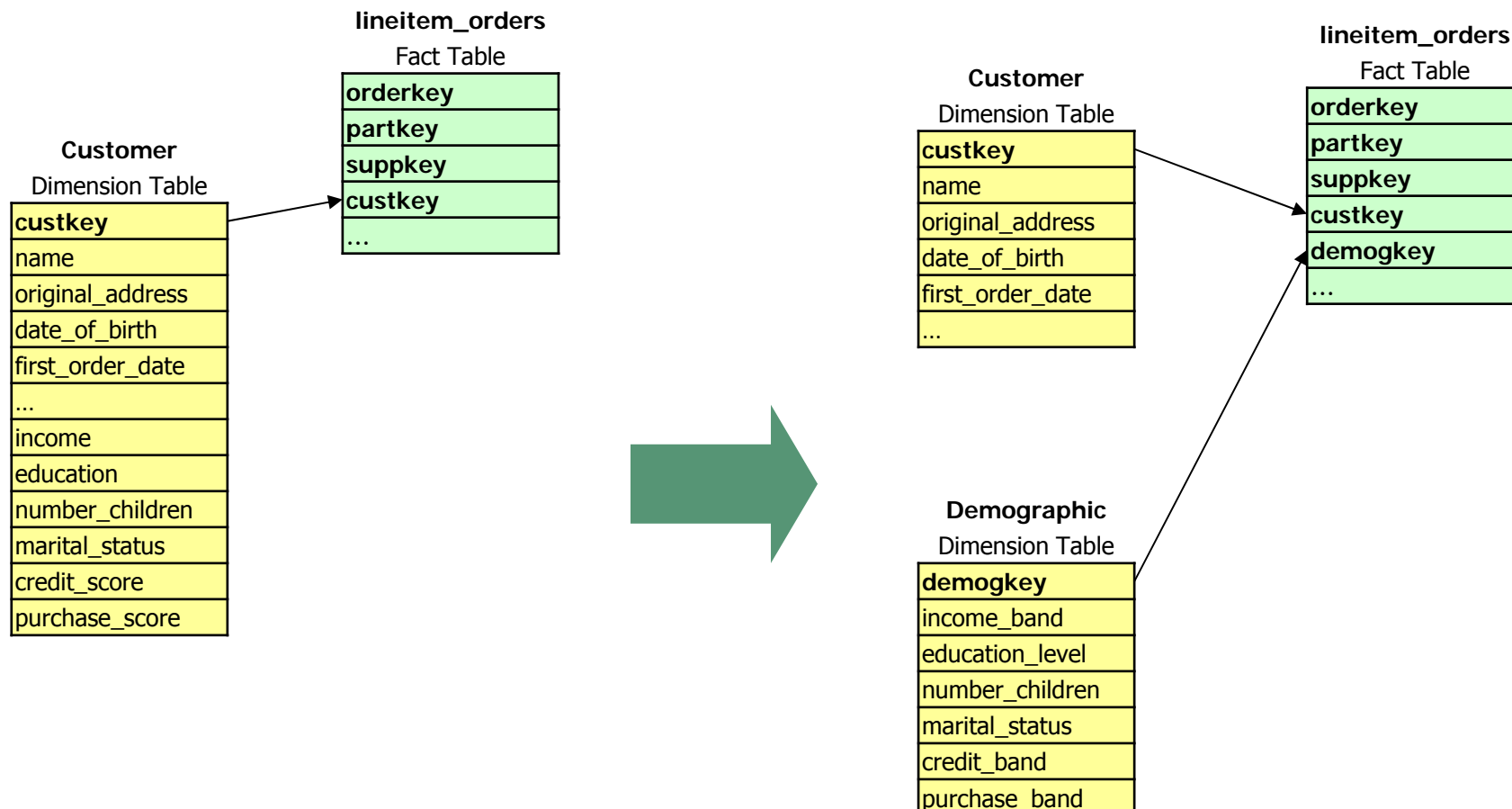
# Large Dimensions with Frequent Changes

- Design goals for this kind of dimensions:
  - Support rapid browsing, e.g. of low cardinality attributes.
  - Support efficient browsing of cross-constrained values in the dimension table.
  - Do not penalize the fact table query for using a large dimension.
  - Find and suppress duplicate entries in the dimension.
  - Do not create additional records to handle the changing dimension problem.




# Large Dimensions with Frequent Changes

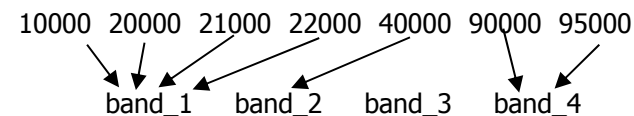
- Modeling approach:  
Break off dimension into several dimension tables.





# Large Dimensions with Frequent Changes

- One (or several) additional dimension table (Demographics) includes all changing attributes.
- Attributes in this new dimension are forced to have a relatively small number of discrete values:
  - e.g., income is mapped to income bands
- The additional dimension table
  - is populated with all possible discrete attribute combinations, and
  - comes with its own surrogate key. 
- The surrogate key of the additional dimension is used in the fact table.
- Changes in the demographics dimension are reflected by entries in the fact table.



# Exercise 5

Data Cleansing and OLAP

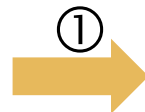
# Task 1

- The Mountain States Health Alliance (MSHA) (see Exercise 2) identified the following issues for data integration and data cleansing:
  - a) Diagnosis information is available on a daily basis for inpatient and a weekly basis for outpatient care.
  - b) All hospitals use the same diagnosis code, but store them in different formats.
  - c) Hospitals use local patient Ids. Laboratories and the imaging center use patient Ids that also encode additional information related to patients (inpatient, outpatient, hospital).
  - d) In the MSIC data, patient Ids are sometimes missing for certain imaging services.
  - e) Only some hospitals deliver information on the expected duration of patients' stay at the hospital.
- What are the data quality issues (if any) related to each of these aspects?
- Discuss advantages and disadvantages of possible steps for data transformation and data cleansing. Which of them can automatically be applied?
- What actions to improve data quality could be taken?

## Task 1 a)

- Diagnosis information is available on a
  - Daily basis for inpatient and a
  - Weekly basis for outpatient care
- Mapping of daily to weekly information is straight-forward ①
- The other way is not possible (except by using some potentially inaccurate default value ②)
- The common ground is always the higher level of abstraction
- Lower level of granularity can be retained where available, but not be used in most analysis
- Change source system to provide more granular information

Day	Diagnosis
01.01.2013	X
15.04.2013	Y
17.09.2013	Z



Week	Diagnosis
01/2013	X
16/2013	Y
38/2013	Z



Day	Diagnosis
31.12.2012	X
15.04.2013	Y
16.09.2013	Z

## Task 1 b)

- All hospitals use the same diagnosis code, but store them in different formats.

Outpatient Care DiagnosisCode
I501
I502
F201
J991
F204
G928

Inpatient Care DiagnosisCode
F20.1
G30.2
F30.2
F10.2
I50.1
I90.1

**Merge  
into  
common  
format**

DiagnosisCode
I50.1
I50.2
F20.1
J99.1
F20.4
G92.8
F20.1
G30.2
F30.2
F10.2
I50.1
I90.1

- Mapping function in DSA (parse codes, identify parts, construct common format)
- Mapping table
- Not really an issue of data quality

## Task 1 c)

- Hospitals use local patient Ids, laboratories and the imaging center use patient Ids that also encode additional information related to patients (inpatient, outpatient, hospital).
- Mapping table/function that maps keys of the source system to the keys in the data warehouse
  - keep data warehouse keys (surrogate keys) independent of source systems
  - if keys are consistent within each source system, there is no important issue of data quality
  - use additional attributes related to patients to identify patients for which data in various source systems exists. Define rules when to consider them as identical.

## Task 1 d)

- In the MSIC data, patient Ids are sometimes missing for certain imaging services.

Outpatient Care ServiceID/PatientID		
1	/	99
2	/	122
3	/	-
4	/	-
6	/	201
9	/	19

### Replace with default value

- Replace missing values with a default value (e.g., -1, 0, INTEGER'LAST etc.)
- + No information lost
- Careless analysis might yield to unexpected results

### Exclude from DWH

- Exclude rows with missing values from the result set altogether
- + Only "complete" information in DWH
- Information is lost

- Find out why Ids are missing and try to ensure that they are available in the source system

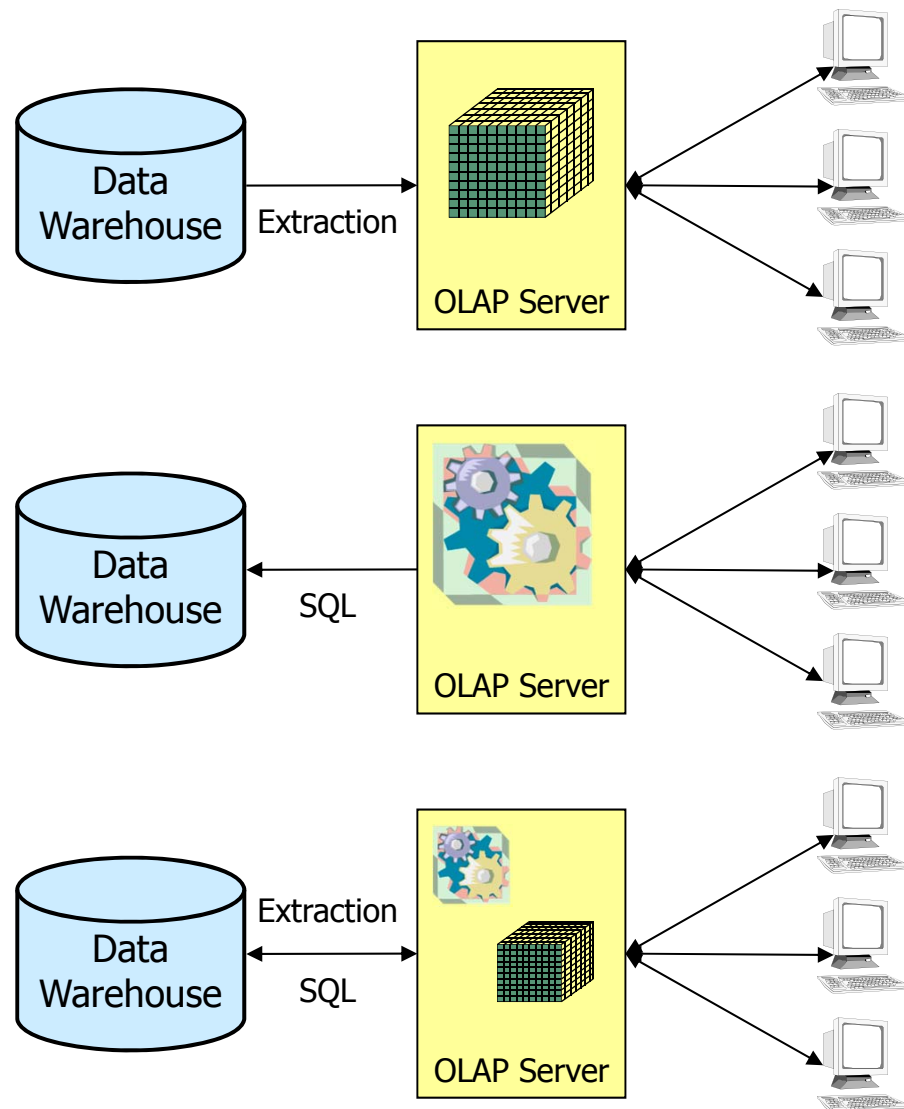
## Task 1 e)

- Only some hospitals deliver information on the expected duration of patients' stay at the hospital.
  - importance:
    - information may be important for hospital management
  - cleansing
    - use data mining to derive expected duration from additional data (patient, diagnoses, ...)



## Task 2

- There are three options for the OLAP architecture: MOLAP, ROLAP and HOLAP. Which of the following statements are true? Argue why the others are wrong.
  1. In MOLAP, ROLAP and HOLAP, the OLAP Servers generate SQL queries.
  2. In MOLAP and HOLAP, the OLAP Server host multidimensional data cubes.
  3. Drill trough is an important operation and hence possible in ROLAP and MOLAP.
  4. The same data warehouse schema might be used to build ROLAP or MOLAP functionality on top of it.
  5. MDX is a query language for ROLAP and MOLAP.




## Task 2

1. In MOLAP, ROLAP and HOLAP, the OLAP Servers generate SQL queries.
  - No, in MOLAP a multidimensional query language like MDX is used in the OLAP server



## Task 2

2. In MOLAP and HOLAP, the OLAP Server host multidimensional data cubes.
- True
  - In MOLAP all requests are processed on these data cubes 
  - IN HOLAP requests may also be processed on the data warehouse

## Task 2

3. Drill through is an important operation and hence possible in ROLAP and MOLAP.
  - No, drill through is associated with HOLAP
  - Only HOLAP allows to go beyond the lowest granularity level defined for the data cubes in the OLAP server

## Task 2

4. The same data warehouse schema might be used to build ROLAP or MOLAP functionality on top of it.
  - True in general, but
  - in ROLAP, the process of generating SQL queries may depend on a certain organization of the data in the data warehouse

## Task 2

5. MDX is a query language for ROLAP and MOLAP.
  - No, a ROLAP server generates SQL queries

## Task 3

- We focus on the following star query:

```
SELECT      month_desc, item_desc, state_desc,
            SUM(reg_sls_dlr), SUM(reg_sls_qty)
FROM        sales s, time t, item I, store o
WHERE       s.CUR_TRN_DATE = t.CUR_TRN_DATE
AND         s.item_key = i.item_key
AND         s.store_key = o.store_key
GROUP BY    month_desc, item_desc, state_desc
```

- Modify the query to reflect the following OLAP operations:
  1. A drill-down in the time dimension to the day level.
  2. A roll-up in the item dimension to the division level.
  3. Slicing in the time dimension (January 2015).
  4. Dicing using all three dimensions (2015, division A, state CA).



## Task 3

- A drill-down in the time dimension to the day level.

```
SELECT      date_desc, item_desc, state_desc,  
            SUM(reg_sls_dlr), SUM(reg_sls_qty)  
FROM        sales s, time t, item I, store o  
WHERE       s.CUR_TRN_DATE = t.CUR_TRN_DATE  
AND         s.item_key = i.item_key  
AND         s.store_key = o.store_key  
GROUP BY    date_desc, item_desc, state_desc
```

## Task 3

- A roll-up in the item dimension to the division level.

```
SELECT      month_desc, division_desc, state_desc,  
            SUM(reg_sls_dlr), SUM(reg_sls_qty)  
FROM        sales s, time t, item I, store o  
WHERE       s.CUR_TRN_DATE = t.CUR_TRN_DATE  
AND         s.item_key = i.item_key  
AND         s.store_key = o.store_key  
GROUP BY    month_desc, division_desc, state_desc
```

## Task 3

- Slicing in the time dimension (January 2015).

```
SELECT      month_desc, item_desc, state_desc,
            SUM(reg_sls_dlr), SUM(reg_sls_qty)
FROM        sales s, time t, item I, store o
WHERE       s.CUR_TRN_DATE = t.CUR_TRN_DATE
AND         s.item_key = i.item_key
AND         s.store_key = o.store_key
AND         t.month_desc = 'January 2015'
GROUP BY    month_desc, item_desc, state_desc
```

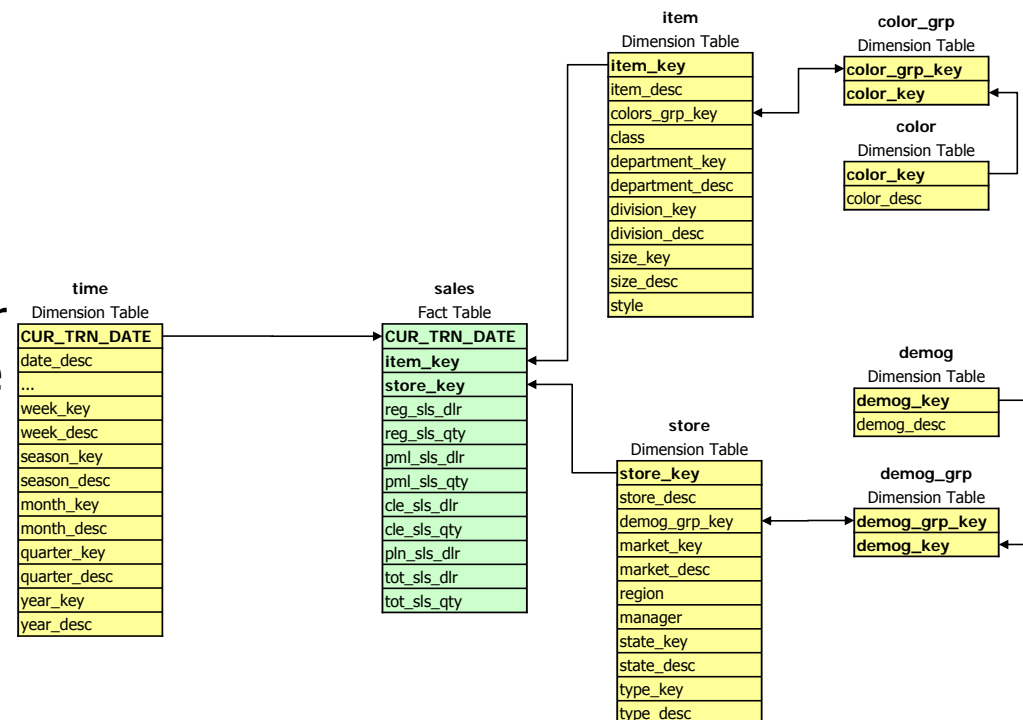
## Task 3

- Dicing using all three dimensions (2015, division A, state CA).

```
SELECT      month_desc, item_desc, state_desc,
            SUM(reg_sls_dlr), SUM(reg_sls_qty)
FROM        sales s, time t, item I, store o
WHERE       s.CUR_TRN_DATE = t.CUR_TRN_DATE
AND         s.item_key = i.item_key
AND         s.store_key = o.store_key
AND         t.year_desc = '2015'
AND         i.division_desc = 'A'
AND         o.state_desc = 'CA'
GROUP BY    month_desc, item_desc, state_desc
```

# Task 4

- The following facts need to be aggregated: reg\_sls\_dlr, reg\_sls\_qty, tot\_sls\_dlr and pln\_sls\_dlr. Create an SQL query that covers this grouping options:
  - year, month, Department, Store
  - year, Department, Store
  - year, Department
  - year
  - total aggregate
- First create a query that returns the total turnover (tot\_sls\_dlr) per year, week, department and store (one result row per combination). Then extend the query in a way, that for each week the aggregated total turnover of the year until this week is returned additionally.



# Query 1

```
SELECT  ti.year_key AS year, ti.month_key AS month,
        it.department_key, st.store_key,
        SUM(reg_sls_dlr) AS reg_sls_dlr,
        SUM(reg_sls_qty) AS reg_sls_qty,
        SUM(tot_sls_dlr) AS tot_sls_dlr,
        SUM(pln_sls_dlr) AS pln_sls_dlr
FROM      DW2.sales sa, DW2.time ti, DW2.item it, DW2.store st
WHERE      ti.cur_trn_date = sa.cur_trn_date
AND        sa.item_key      = it.item_key
AND        sa.store_key     = st.store_key
GROUP BY   GROUPING SETS(
    (ti.year_key, it.department_key, st.store_key, ti.month_key),
    (ti.year_key, it.department_key, st.store_key),
    (ti.year_key, it.department_key),
    (ti.year_key),
    ()
);
```

# Query 1

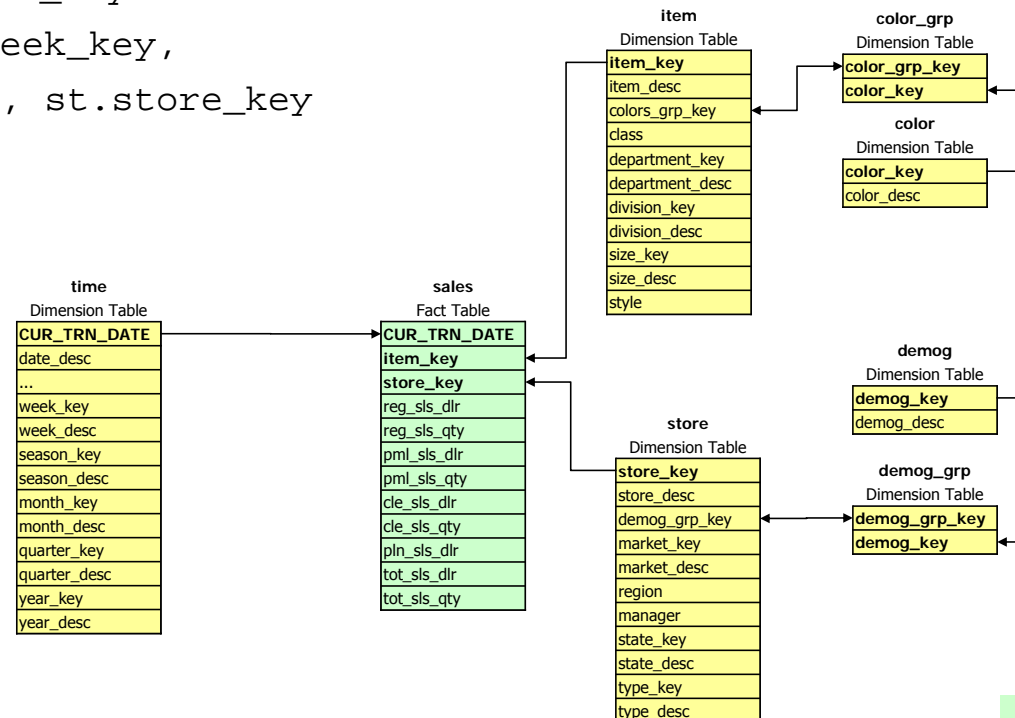
```
SELECT  ti.year_key AS year, ti.month_key AS month,
        it.department_key, st.store_key,
        SUM(reg_sls_dlr) AS reg_sls_dlr,
        SUM(reg_sls_qty) AS reg_sls_qty,
        SUM(tot_sls_dlr) AS tot_sls_dlr,
        SUM(pln_sls_dlr) AS pln_sls_dlr
FROM      DW2.sales sa, DW2.time ti, DW2.item it, DW2.store st
WHERE     ti.cur_trn_date = sa.cur_trn_date
AND       sa.item_key      = it.item_key
AND       sa.store_key     = st.store_key
GROUP BY
ROLLUP(
    ti.year_key, it.department_key, st.store_key, ti.month_key);
```

## Query 2

```

SELECT    it.department_key, st.store_key,
          ti.year_key AS year, ti.week_key AS week,
          SUM(tot_sls_dlr) AS tot_sls_dlr

FROM      DW2.sales sa, DW2.time ti, DW2.item it, DW2.store st
WHERE     ti.cur_trn_date = sa.cur_trn_date
and       sa.item_key = it.item_key
and       sa.store_key = st.store_key
GROUP BY  ti.year_key, ti.week_key,
          it.department_key, st.store_key
  
```





## Query 2

DEPARTMENT_KEY	STORE_KEY	YEAR	WEEK	TOT_SLS_DLR
1	1	1999	48	4657.00
1	1	1999	49	3663.00
1	1	1999	50	2354.00
1	1	1999	51	2136.00
1	1	1999	52	1715.00
1	1	2000	48	4619.00
1	1	2000	49	4216.00
1	1	2000	50	2562.00
1	1	2000	51	2396.00
1	1	2000	52	3200.00
1	1	2001	1	503.00
1	1	2001	48	2670.00
1	1	2001	49	5742.00
1	1	2001	50	2897.00
1	1	2001	51	2184.00

## Query 2

DEPARTMENT_KEY	STORE_KEY	YEAR	WEEK	TOT_SLS_DLR	TOT_SLS_DLR_AGR
1	1	1999	48	4657.00	4657.00
1	1	1999	49	3663.00	8320.00
1	1	1999	50	2354.00	10674.00
1	1	1999	51	2136.00	12810.00
1	1	1999	52	1715.00	14525.00
1	1	2000	48	4619.00	4619.00
1	1	2000	49	4216.00	8835.00
1	1	2000	50	2562.00	11397.00
1	1	2000	51	2396.00	13793.00
1	1	2000	52	3200.00	16993.00
1	1	2001	1	503.00	503.00
1	1	2001	48	2670.00	3173.00
1	1	2001	49	5742.00	8915.00
1	1	2001	50	2897.00	11812.00
1	1	2001	51	2184.00	13996.00

## Query 2

```
SELECT department_key, store_key, year, week, tot_sls_dlr,
       SUM(tot_sls_dlr) OVER (
         PARTITION BY department_key,store_key,year
         ORDER BY week
         ROWS 53 PRECEDING) AS tot_sls_dlr_agr
FROM (  SELECT  ti.year_key AS year, ti.week_key AS week,
              it.department_key, st.store_key,
              SUM(tot_sls_dlr) AS tot_sls_dlr
        FROM    DW2.sales sa, DW2.time ti, DW2.item it, DW2.store st
        WHERE   ti.cur_trn_date = sa.cur_trn_date
        and     sa.item_key = it.item_key
        and     sa.store_key = st.store_key
        GROUP BY ti.year_key, ti.week_key,
              it.department_key, st.store_key
        ) AS t;
```

## Task 5

- Starting point is a multidimensional partitioning of an n-dimensional data cube with m partitions  $b_1$  to  $b_m$ :

$$b_1 = [l_{1,1} : u_{1,1}; \dots; l_{1,n} : u_{1,n}] \dots b_m = [l_{m,1} : u_{m,1}; \dots; l_{m,n} : u_{m,n}]$$

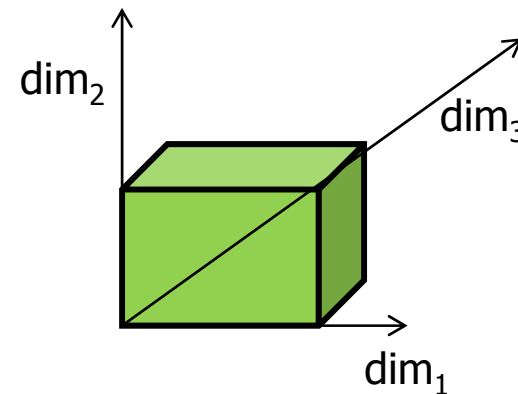
- Provide a generic solution for calculating the needed storage for partition  $b_i$  of a data cube using multidimensional or relational storage. Consider the filling degree  $\delta$ .
- Describe until which filling degree the multidimensional storage is superior compared to the relational storage with respect to the needed storage. Calculate the results for 2, 3 and 4 dimensions. Assume that we need 16 bytes to store one fact value and 8 bytes to store one dimension attribute.

# Multidimensional vs. relational storage

$$b_i = [l_{i,1}:u_{i,1}; \dots; l_{i,n}:u_{i,n}]$$



- $S_q$ : number of possible attribute values in dim.  $q$  (in partition  $b_i$ )
- $C$ : storage needed for one fact value (= 16)
- $D_p$ : storage needed to store an attribute value in dimension  $p$  (=8)

customer_id	product_id	time_id	quantity
100	200	01.01.2011	1

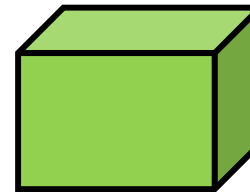


# Multidimensional vs. relational storage

$$b_i = [l_{i,1}:u_{i,1}; \dots; l_{i,n}:u_{i,n}]$$

- $S_q$ : number of possible attribute values in dim.  $q$  (in partition  $b_i$ )
- $C$ : storage needed for one fact value (= 16) 
- $D_p$ : storage needed to store an attribute value in dimension  $p$  (=8) 

customer_id	product_id	time_id	quantity
100	200	01.01.2011	1



Relational storage for partition  $b_i$ :

$$\delta \cdot \prod_{q=1}^n S_q \cdot \left( C + \sum_{p=1}^n D_p \right)$$

Multidimensional storage for partition  $b_i$ :

$$C \cdot \prod_{q=1}^n S_q$$

# Multidimensional vs. relational storage

Multidimensional storage is better, if:

$$\delta \cdot \prod_{q=1}^n S_q \cdot \left( C + \sum_{p=1}^n D_p \right) > C \cdot \prod_{q=1}^n S_q \quad \text{i.e.} \quad \delta > \frac{C}{C + \sum_{p=1}^n D_p}$$

In 2-dimensional cases:

$$\delta \cdot \prod_{q=1}^2 S_q \cdot \left( 16 + \sum_{p=1}^2 8 \right) > 16 \cdot \prod_{q=1}^2 S_q \quad \text{i.e.} \quad \delta > \frac{16}{32} = 0,5$$

In 3-dimensional cases:

$$\delta \cdot \prod_{q=1}^3 S_q \cdot \left( 16 + \sum_{p=1}^3 8 \right) > 16 \cdot \prod_{q=1}^3 S_q \quad \text{i.e.} \quad \delta > \frac{16}{40} = 0,4$$

In 4-dimensional cases:

$$\delta \cdot \prod_{q=1}^4 S_q \cdot \left( 16 + \sum_{p=1}^4 8 \right) > 16 \cdot \prod_{q=1}^4 S_q \quad \text{i.e.} \quad \delta > \frac{16}{48} = 0,33$$

# Exercise 6

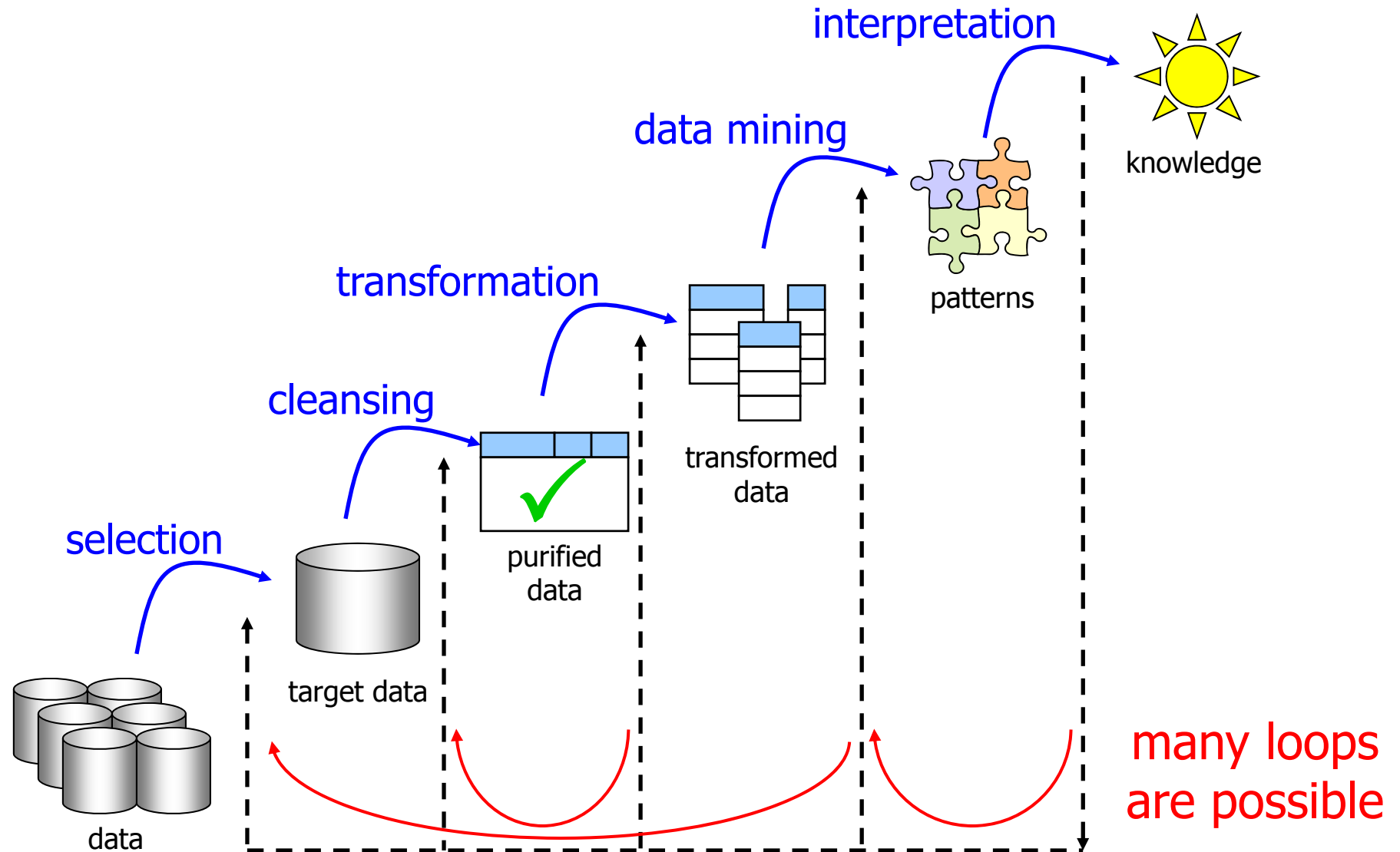
Data Mining



# Task 1

- Name the phases of the knowledge discovery process and explain the purpose of each of the steps in one or two sentences.

# Knowledge Discovery Process



## Task 2

- Mention and explain the three phases of data mining. Which of them are applicable to all four data mining techniques? Why are some phases not relevant for some data mining techniques?

# Data Mining Phases

Data mining technique	Training phase	Test phase	Application phase
Classification	✓	✓	✓
Regression	✓	✓	✓
Clustering	✓	✗	✓
Association rule discovery	✓	✗	✗

Source: ISO/IEC 13249-6:2002 Information technology -- Database languages -- SQL multimedia and application packages -- Part 6: Data mining

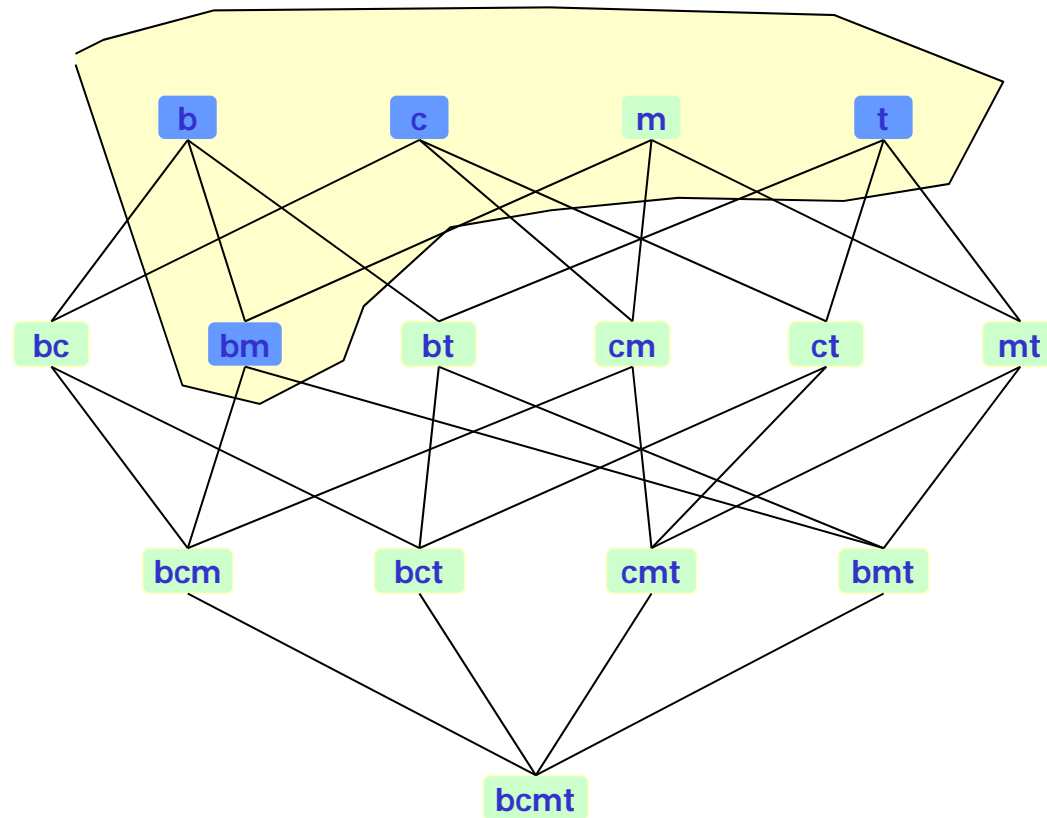
## Task 3

- Assume we are trying to identify frequent itemsets in transactions that include some of the items beer, cheese, tomatoes and milk. We already know that
  - the following itemsets are frequent: {beer}, {cheese}, {tomatoes}, {beer, milk}
  - the following itemset is not frequent: {cheese, tomatoes}
- Which of the following facts can we directly derive from this knowledge using the apriori property?
  1. Itemsets {beer, cheese, tomatoes}, {cheese, milk, tomatoes}, {beer, cheese, milk, tomatoes} are not frequent
  2. Itemset {milk} is frequent
  3. Itemset {beer, cheese} is frequent

# Apriori Property

Database D

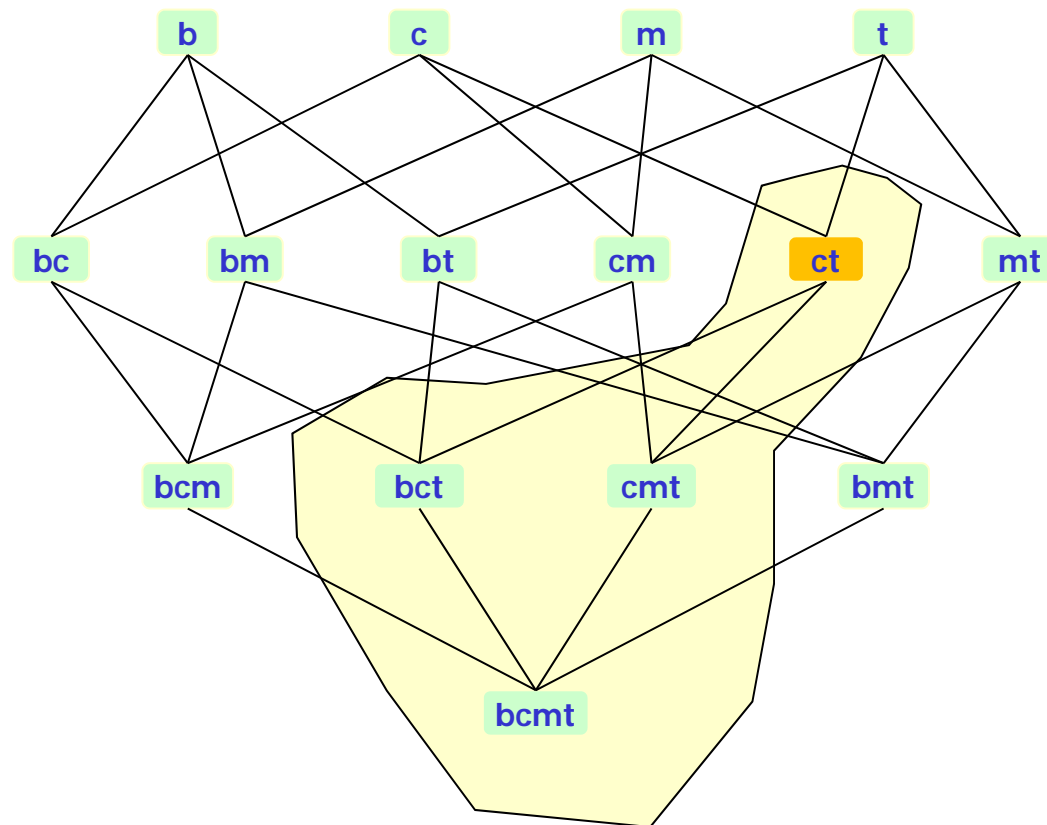
tid	item
101	beer
101	cheese
102	beer
102	milk
102	tomatoes
103	cheese



# Apriori Property

Database D

tid	item
101	beer
101	cheese
102	beer
102	milk
102	tomatoes
103	cheese



## Task 4

- The following table contains a set of transactions:

TID	Itemset
100	A, B, E
200	B, D
300	B, C
400	A, B, D
500	A, C
600	B, C
700	A, C
800	A, B, C, E
900	A, B, C
1000	A, B, C, F

- Use the Apriori algorithm to calculate the frequent itemsets (minimum support = 0.2) on basis of these transactions. Describe each step in detail and provide the intermediate result for each iteration. What is the stop criteria of the algorithm?



TID	Itemset
100	A, B, E
200	B, D
300	B, C
400	A, B, D
500	A, C
600	B, C
700	A, C
800	A, B, C, E
900	A, B, C
1000	A, B, C, F

$C[1] + F[1]$


Itemset	Support Count
{A}	7
{B}	8
{C}	7
{D}	2
{E}	2
{F}	1

$C[2] + F[2]$

Itemset	Support Count
{A, B}	5
{A, C}	5
{A, D}	1
{A, E}	2
{B, C}	5
{B, D}	2
{B, E}	2
{C, D}	0
{C, E}	1
{D, E}	0

$C[3] + F[3]$

Itemset	Support
{A, B, C}	3
{A, B, E}	2
{A, C, E}	
{B, C, D}	
{B, C, E}	
{B, D, E}	

 no candidates due to infrequent subset

$\{A, B, C, E\}$  hasInfrequentSubset  $\{B, C, E\}$

$C[4] = \emptyset, F[4] = \emptyset$

## Task 5

- The algorithm ap-genrules derives association rules (see next slide)
  1. Use this algorithm on the example of the lecture. (minimum confidence = 0,5)
  2. Compare the behavior of the algorithm with the simple algorithm from the lecture. The frequent itemsets in the adjacent table are the starting point. Consider only rules that can be created from itemset {A, B, C, D, E} (minimum confidence = 0.4).
  3. Also indicate the lift factor for the created rules. Assume that you have 50 transactions.

Element of F	count
{A}, {B}, {C}, {D}, {E}	9
{A, B}, {A, C}, ..., {D, E}	8
{A, B, C}, {A, B, D}, {A, B, E}, ..., {B, D, E}, {C, D, E}	7
{B, C, D, E}	6
{A, B, D, E}	6
{A, C, D, E}	5
{A, B, C, E}	5
{A, B, C, D}	6
{A, B, C, D, E}	2

# Algorithm ap-genrules

- Precondition: all frequent itemsets have been computed
- each itemset has two attributes:
  - count: „support“ as an integer
  - size: cardinality, i.e. number of items in itemset

```

for each frequent k-itemset  $f \in F$ ,  $k \geq 2$  do
     $H_1 = \{R \mid L \rightarrow R \text{ is association rule with } |R|=1 \text{ and } f = L \cup R\}$ 
    call ap-genrules( $f$ ,  $H_1$ );
end;

ap-genrules(Itemset  $f_k$ , Itemsets  $right_m$ ) begin
    if ( $k > m+1$ ) then begin
         $right_{m+1} = \text{generateCandidates}(m, right_m)$ 
        for each  $r_{m+1} \in right_{m+1}$  do begin
            confidence =  $f_k.count / (f_k - r_{m+1}).count$ ;
            if confidence  $\geq \text{minConfidence}$  then
                rules.add( $f_k - r_{m+1}$ ,  $r_{m+1}$ );
            else
                delete  $r_{m+1}$  from  $right_{m+1}$ ;
            end;
        end;
        end;
        call ap-genrules( $f_k$ ,  $right_{m+1}$ );
    end;
end;

```

# Algorithm generateRules

- Precondition: all frequent itemsets have been computed
- each itemset has two attributes:
  - count: „support“ as an integer
  - size: cardinality, i.e. number of items in itemset

```
for each frequent k-itemset  $f \in F$  where  $k \geq 2$  do
    generateRules(f, f);
// Computes all rules ( $g \rightarrow f \setminus g$ ) for all  $g \subset f$ .
generateRules(Itemset f, Itemset left) begin
     $A = \{a \subset \text{left} \mid a \text{ is of size: } (\text{left.size}-1) \}$ ;
    for each  $a \in A$  do begin
        confidence = f.count / a.count;
        if confidence  $\geq$  minConfidence then begin
            rules.add(a,  $f \setminus a$ );           // set of result association rules
            if (left.size-1 > 1) then
                generateRules(f, a);
        end;
    end;
end;
```

# Example using generateRules

itemset	count
{3}	5
{4}	4
{5}	2
{3,4}	3
{3,5}	2
{4,5}	2
{3,4,5}	2

We restrict ourselves to itemset {3,4,5}.

Minimum confidence =  $\frac{1}{2} = 50\%$ .

generateRules({3,4,5}, {3,4,5}):

$A = \{\{3,4\}, \{3,5\}, \{4,5\}\}$ .

$a = \{3,4\}$ :  $c(a) = \frac{2}{3} \geq \frac{1}{2} \rightarrow \{3,4\} \rightarrow \{5\}$  is AR

$\rightarrow$  generateRules({3,4,5}, {3,4}):

$A = \{\{3\}, \{4\}\}$ .

$a = \{3\}$ :  $c(a) = \frac{2}{5} < \frac{1}{2}$ .

$a = \{4\}$ :  $c(a) = \frac{2}{4} \geq \frac{1}{2} \rightarrow \{4\} \rightarrow \{3,5\}$  is AR

$a = \{3,5\}$ :  $c(a) = \frac{2}{2} \geq \frac{1}{2} \rightarrow \{3,5\} \rightarrow \{4\}$  is AR

$\rightarrow$  generateRules({3,4,5}, {3,5}):

$A = \{\{3\}, \{5\}\}$ .

$a = \{5\}$ :  $c(a) = \frac{2}{2} \geq \frac{1}{2} \rightarrow \{5\} \rightarrow \{3,4\}$  is AR

... (as before)

$a = \{4,5\}$ :  $c(a) = \frac{2}{2} \geq \frac{1}{2} \rightarrow \{4,5\} \rightarrow \{3\}$  is AR

$\rightarrow$  generateRules({3,4,5}, {4,5}):

$A = \{\{4\}, \{5\}\}$ .

... (as before)



Thus, the following association rules can be derived from frequent itemset {3,4,5}:

$\{4\} \rightarrow \{3,5\}$

$\{5\} \rightarrow \{3,4\}$

$\{3,4\} \rightarrow \{5\}$

$\{3,5\} \rightarrow \{4\}$

$\{4,5\} \rightarrow \{3\}$

## Example using ap-genrules

itemset	count
{3}	5
{4}	4
{5}	2
{3,4}	3
{3,5}	2
{4,5}	2
{3,4,5}	2

We restrict ourselves to itemset  $\{3,4,5\}$ ,  $k=3$ .

Minimum confidence =  $\frac{1}{2} = 50\%$ .

determine  $H_1$ :

$\{3, 4\} \rightarrow \{5\} : c = 2/3 > \frac{1}{2} \rightarrow \{3, 4\} \rightarrow \{5\}$  is AR

$\{3, 5\} \rightarrow \{4\} : c = 2/2 > \frac{1}{2} \rightarrow \{3, 5\} \rightarrow \{4\}$  is AR

$\{4, 5\} \rightarrow \{3\} : c = 2/2 > \frac{1}{2} \rightarrow \{4, 5\} \rightarrow \{3\}$  is AR

$H_1 = \{\{3\}, \{4\}, \{5\}\} = \text{right}_1$

$\text{ap\_genrules}(\{3,4,5\}, H_1)$

$\text{right}_2 = \text{generate\_candidates}(1, \text{right}_1) = \{\{3,4\}, \{4,5\}, \{3,5\}\}$

$\{5\} \rightarrow \{3, 4\} : c = 2/2 > \frac{1}{2} \rightarrow \{5\} \rightarrow \{3, 4\}$  is AR

$\{3\} \rightarrow \{4, 5\} : c = 2/5 < \frac{1}{2} \rightarrow \text{right}_2 = \text{right}_2 - \{4, 5\}$

$\{4\} \rightarrow \{3, 5\} : c = 2/4 \geq \frac{1}{2} \rightarrow \{4\} \rightarrow \{3, 5\}$  is AR

$m=2: k = 3 = 2+1 = m+1 \rightarrow \text{end}$



Thus, the following association rules can be derived from frequent itemset  $\{3,4,5\}$ :

$\{4\} \rightarrow \{3,5\}$

$\{5\} \rightarrow \{3,4\}$

$\{3,4\} \rightarrow \{5\}$

$\{3,5\} \rightarrow \{4\}$

$\{4,5\} \rightarrow \{3\}$

# generateRules

itemset	count
{A}, ..., {E}	9
{A,B}, ..., {D,E}	8
{A,B,C}, ..., {C,D,E}	7
{B,C,D,E}	6
{A,B,D,E}	6
{A,C,D,E}	5
{A,B,C,E}	5
{A,B,C,D}	6
{A,B,C,D,E}	2

We restrict ourselves to itemset  $\{A,B,C,D,E\}$

Minimum confidence = 0.4

`generateRules( $\{A,B,C,D,E\}$ ,  $\{A,B,C,D,E\}$ )`

$A = \{\{A,B,C,D\}, \{A,B,D,E\}, \{B,C,D,E\}, \{A,C,D,E\}, \{A,B,C,E\}\}$

$\{A,B,C,D\} \rightarrow \{E\} : c=2/6 < 0.4 \rightarrow$  no AR

$\{A,B,D,E\} \rightarrow \{C\} : c=2/6 < 0.4 \rightarrow$  no AR

$\{B,C,D,E\} \rightarrow \{A\} : c=2/6 < 0.4 \rightarrow$  no AR

$\{A,C,D,E\} \rightarrow \{B\} : \text{see next slide}$

$\{A,B,C,E\} \rightarrow \{D\} : \text{see next slide}$

# generateRules

itemset	count
{A}, ..., {E}	9
{A,B}, ..., {D,E}	8
{A,B,C}, ..., {C,D,E}	7
{B,C,D,E}	6
{A,B,D,E}	6
{A,C,D,E}	5
{A,B,C,E}	5
{A,B,C,D}	6
{A,B,C,D,E}	2

$\{A,C,D,E\} \rightarrow \{B\} : c=2/5 \geq 0.4 \rightarrow \{A,C,D,E\} \rightarrow \{B\}$  is AR

generateRules( $\{A,B,C,D,E\}$ ,  $\{A,C,D,E\}$ )

$\{A,C,D\} \rightarrow \{B,E\} : c=2/7 < 0.4 \rightarrow$  no AR

$\{A,D,E\} \rightarrow \{B,C\} : c=2/7 < 0.4 \rightarrow$  no AR

$\{A,C,E\} \rightarrow \{B,D\} : c=2/7 < 0.4 \rightarrow$  no AR

$\{C,D,E\} \rightarrow \{A,B\} : c=2/7 < 0.4 \rightarrow$  no AR

$\{A,B,C,E\} \rightarrow \{D\} : c=2/5 \geq 0.4 \rightarrow \{A,B,C,E\} \rightarrow \{D\}$  is AR

generateRules( $\{A,B,C,D,E\}$ ,  $\{A,B,C,E\}$ )

$\{A,B,C\} \rightarrow \{D, E\} : c=2/7 < 0.4 \rightarrow$  no AR

$\{A,B,E\} \rightarrow \{C,D\} : c=2/7 < 0.4 \rightarrow$  no AR

$\{A,C,E\} \rightarrow \{B,D\} : c=2/7 < 0.4 \rightarrow$  no AR

$\{B,C,E\} \rightarrow \{A,D\} : c=2/7 < 0.4 \rightarrow$  no AR

Thus, the following association rules can be derived from frequent itemset  $\{A,B,C,D,E\}$ :

$\{A,C,D,E\} \rightarrow \{B\}$

$\{A,B,C,E\} \rightarrow \{D\}$

11 comparisons of confidence values



# ap\_genrules

itemset	count
{A}, ..., {E}	9
{A,B}, ..., {D,E}	8
{A,B,C}, ..., {C,D,E}	7
{B,C,D,E}	6
{A,B,D,E}	6
{A,C,D,E}	5
{A,B,C,E}	5
{A,B,C,D}	6
{A,B,C,D,E}	2

We restrict ourselves to itemset  $\{A,B,C,D,E\}$

Minimum confidence = 0.4

determine  $H_1$ :

$\{A,B,C,D\} \rightarrow \{E\} : c=2/6 < 0.4 \rightarrow$  no AR

$\{A,B,C,E\} \rightarrow \{D\} : c=2/5 \geq 0.4 \rightarrow \{A,B,C,E\} \rightarrow \{D\}$  is AR

$\{A,B,D,E\} \rightarrow \{C\} : c=2/6 < 0.4 \rightarrow$  no AR

$\{A,C,D,E\} \rightarrow \{B\} : c=2/5 \geq 0.4 \rightarrow \{A,C,D,E\} \rightarrow \{B\}$  is AR

$\{B,C,D,E\} \rightarrow \{A\} : c=2/6 < 0.4 \rightarrow$  no AR

$H_1 = \{\{B\}, \{D\}\} = \text{right}_1$

$\text{ap\_genrules}(\{A,B,C,D,E\}, H_1)$

# ap\_genrules

itemset	count
{A}, ..., {E}	9
{A,B}, ..., {D,E}	8
{A,B,C}, ..., {C,D,E}	7
{B,C,D,E}	6
{A,B,D,E}	6
{A,C,D,E}	5
{A,B,C,E}	5
{A,B,C,D}	6
{A,B,C,D,E}	2

ap\_genrules({A,B,C,D,E},  $H_1$ )

right<sub>2</sub> = generate\_candidates(1, right<sub>1</sub>) = {{B, D}}

{A,C,E} → {B,D} :  $c=2/7 < 0.4 \rightarrow$  no AR

ap\_genrules({A,B,C,D,E}, {B, D})

right<sub>3</sub> = generate\_candidates(2, right<sub>2</sub>) = {}

no further calls to ap\_genrules



Thus, the following association rules can be derived from frequent itemset {A,B,C,D,E}:

{A,C,D,E} → {B}

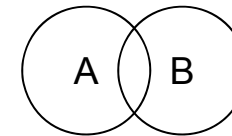
{A,B,C,E} → {D}

6 comparisons of confidence values

# Beyond Support & Confidence: Interest

- One can define further measures to find „interesting“ AR
- Two events A and B are independent if the predicate  $p(A \wedge B) = p(A) \times p(B)$  holds, otherwise they are correlated
- Interest of itemsets  $\{A, B\}$ :  

$$s(\{A, B\}) / (s(\{A\}) \times s(\{B\}))$$
- It is a measure of correlation between events
  - interest = 1: A and B are independent events
  - interest < 1: A and B are negatively correlated
  - interest > 1: A and B are positively correlated
- Sometimes, interest is also called “surprise factor” or “lift factor”



	coffee	not coffee	$\Sigma$
tea	20	5	25
not tea	70	5	75
$\Sigma$	90	10	100

example: AR  $\{\text{tea}\} \rightarrow \{\text{coffee}\}$

support =  $20/100 = 0.2$

confidence =  $(20/100) / (25/100) = 0.8$

interest =  $0.2 / (0.25 \times 0.9) = 0.89$ ,

i.e. interest(drink tea  $\wedge$  drink coffee) is negatively correlated, i.e. it is rather unlikely that you start drinking coffee if you are already a tea worshipper

# lift factor

itemset	count
{A}, ..., {E}	9
{A,B}, ..., {D,E}	8
{A,B,C}, ..., {C,D,E}	7
{B,C,D,E}	6
{A,B,D,E}	6
{A,C,D,E}	5
{A,B,C,E}	5
{A,B,C,D}	6
{A,B,C,D,E}	2



Thus, the following association rules can be derived from frequent itemset {A,B,C,D,E}:

$\{A,C,D,E\} \rightarrow \{B\}$

$\{A,B,C,E\} \rightarrow \{D\}$

lift: (50 transactions)

$$s(\{A,C,D,E\}) = 5/50 = 0.1$$

$$s(\{B\}) = 9/50 = 0.18$$

$$s(\{A,B,C,D,E\}) = 2/50 = 0.04$$

$$\text{Lift}(\{A,C,D,E\} \rightarrow \{B\})$$

$$= 0.04/(0.1*0.18)$$

$$= 22.2$$

lift: (50 transactions)

$$s(\{A,B,C,E\}) = 5/50 = 0.1$$

$$s(\{D\}) = 9/50 = 0.18$$

$$s(\{A,B,C,D,E\}) = 2/50 = 0.04$$

$$\text{Lift}(\{A,B,C,E\} \rightarrow \{D\})$$

$$= 0.04/(0.1*0.18)$$

$$= 22.2$$

## Task 6

- The following information on the different stores of a department store chain is available. The attributes area, staff members and turnover have already been normalized to get the same range of values for all dimensions.

1. Divide the stores into three clusters by means of the K-Means algorithm (Euclidian distance) considering the attributes area, staff members and turnover. Initially, the stores with the IDs 701, 702 and 703 need to be chosen as prototypes/centres of the three clusters.
2. Execute the clustering again considering this time only staff members and turnover. Compare the amount of needed iterations and how the result depends on the initial values. Start the algorithm first with the stores 701, 702 and 703 as cluster centres. Try a second run with the initial stores 702, 705 and 706.

Store ID	area	staff	turnover	Store type
701	17	18	17	A
702	25	14	1,12	B
703	9	23	7	A
704	20	30	34	A
705	10	11	3,5	B
706	15	15	9	A

# K-Means Algorithm

- Given:
  - k: number of clusters
  - a “database” containing n objects

```
k_means( ) begin  
  arbitrarily choose k objects as the initial cluster centers;  
  repeat  
    assign each object to the cluster to which the object is  
    the most similar based on the mean value of the objects  
    of the cluster;  
    calculate the mean value of the objects for each cluster;  
  until no change;  
end;
```

Store ID	area	staff	turnover	Store type
701	17	18	17	A
702	25	14	1,12	B
703	9	23	7	A
704	20	30	34	A
705	10	11	3,5	B
706	15	15	9	A

#	Area	staff	turnover
701	17,00	18,00	17,00
702	25,00	14,00	1,12
703	9,00	23,00	7,00
704	20,00	30,00	34,00
705	10,00	11,00	3,50
706	15,00	15,00	9,00
M1=701	17,00	18,00	17,00
M2=702	25,00	14,00	1,12
M3=703	9,00	23,00	7,00
M1-1	17,33	21,00	20,00
M2-1	25,00	14,00	1,12
M3-1	9,50	17,00	5,25
M1-2	18,50	24,00	25,50
M2-2	25,00	14,00	1,12
M3-2	11,33	16,33	6,50

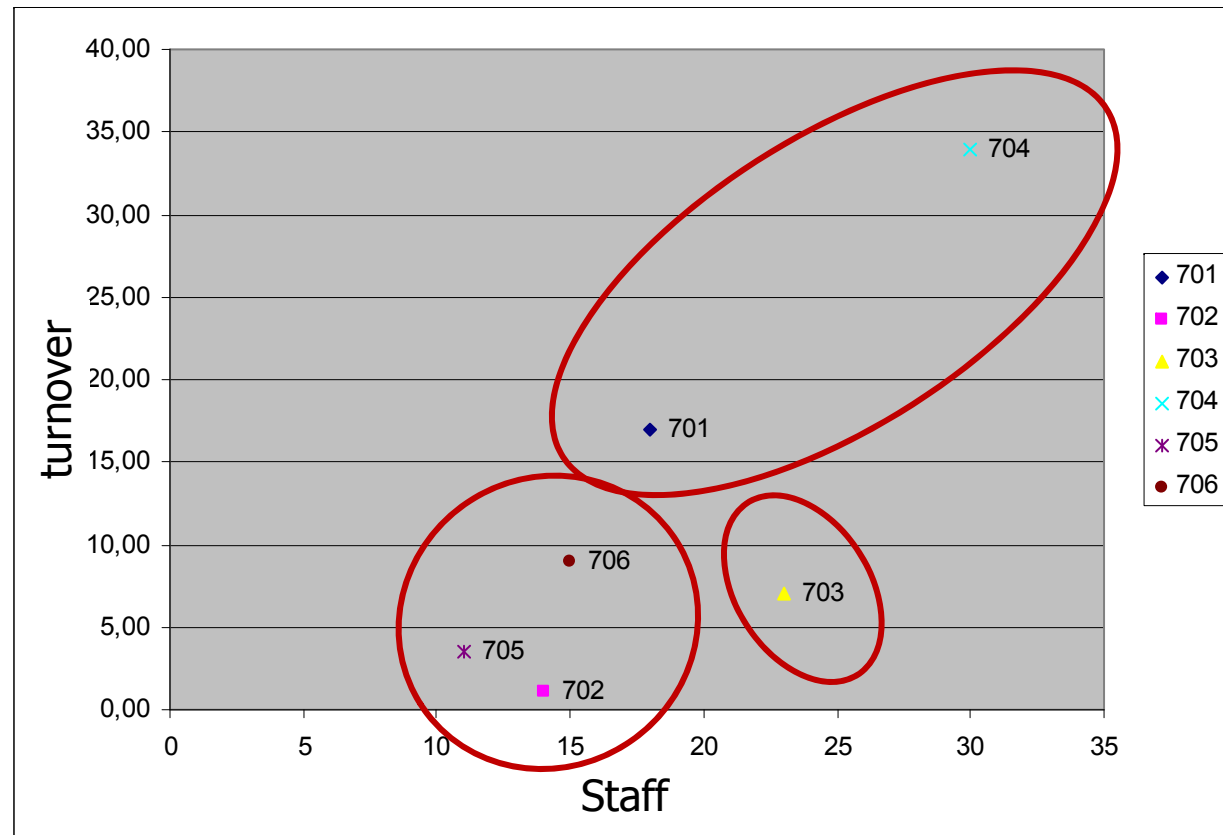
Distances from

M1	M2	M3		M1-1	M2-1	M3-1		M1-1	M2-1	M3-1
0,00	18,23	13,75		4,26	18,23	13,98		10,51	18,23	12,05
18,23	0,00	19,28		21,55	0,00	16,32		27,14	0,00	14,87
13,75	19,28	0,00		15,57	19,28	6,27		20,82	19,28	7,08
21,02	36,91	29,98		16,86	36,91	33,25		10,51	36,91	31,91
16,74	15,48	12,54		20,64	15,48	6,27		26,93	15,48	6,26
8,77	12,77	10,20		12,75	12,77	6,95		19,12	12,77	4,63

#	Staff	turnover
701	18,00	17,00
702	14,00	1,12
703	23,00	7,00
704	30,00	34,00
705	11,00	3,50
706	15,00	9,00
M1=701	18,00	17,00
M2=702	14,00	1,12
M3=703	23,00	7,00
M1-1	24,00	25,50
M2-1	13,33	4,54
M3-1	23,00	7,00

Distances from

M1	M2	M3		M1-1	M2-1	M3-1
0,00	16,38	11,18		10,40	13,31	11,18
16,38	0,00	10,75		26,35	3,48	10,75
11,18	10,75	0,00		18,53	9,97	0,00
20,81	36,57	27,89		10,40	33,85	27,89
15,21	3,83	12,50		25,55	2,55	12,50
8,54	7,94	8,25		18,79	4,76	8,25



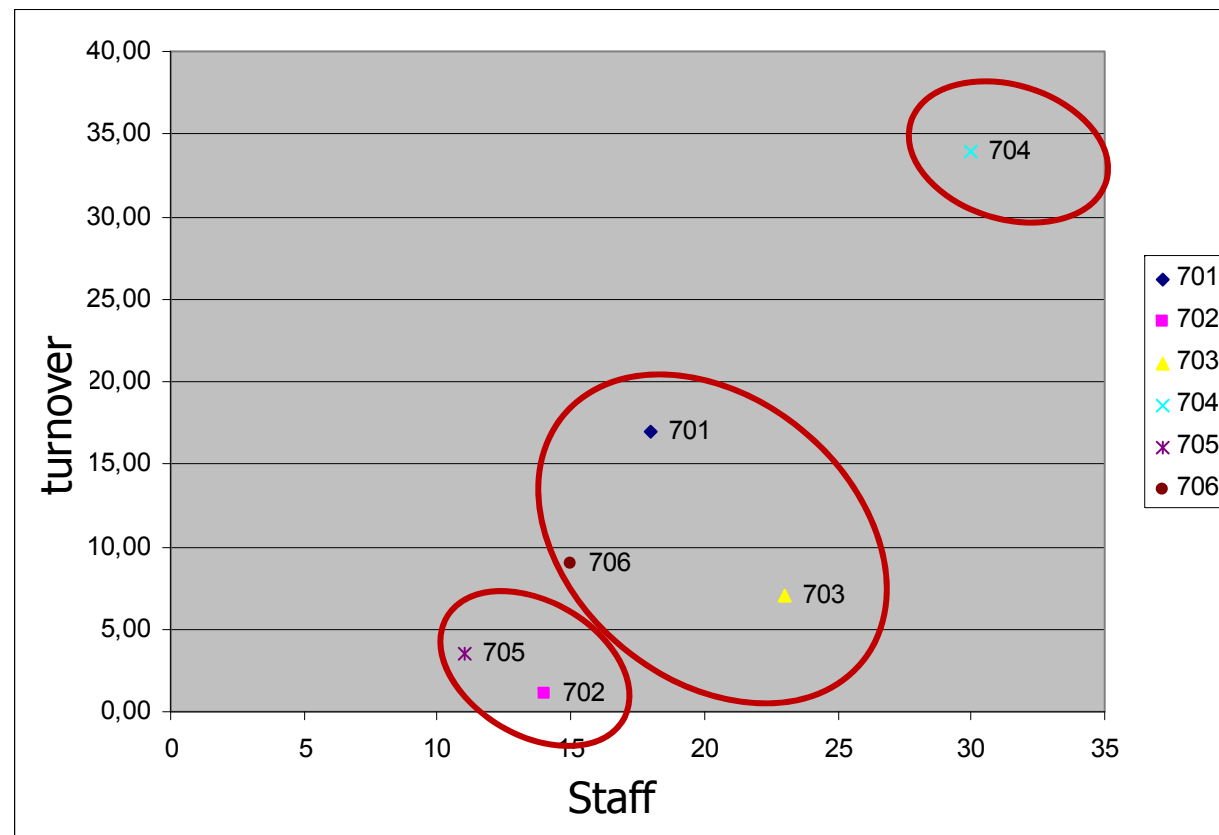


Staff - turnover

#	Staff	turnover
701	18,00	17,00
702	14,00	1,12
703	23,00	7,00
704	30,00	34,00
705	11,00	3,50
706	15,00	9,00
M1=702	14,00	1,12
M2=705	11,00	3,50
M3=706	15,00	9,00
M1-1	14,00	1,12
M2-1	11,00	3,50
M3-1	21,50	16,75
M1-2	14,00	1,12
M2-2	13,00	6,25
M3-2	23,67	19,33
M1-3	14,00	1,12
M2-3	16,33	6,50
M3-3	24,00	25,50
M1-4	12,50	2,31
M2-4	19,00	8,00
M3-4	24,00	25,50
M1-5	12,50	2,31
M2-5	18,67	11,00
M3-5	30,00	34,00

Distances from

M1	M2	M3		M1-1	M2-1	M3-1		M1-2	M2-2	M3-2
16,38	15,21	8,54		16,38	15,21	3,51		16,38	11,86	6,13
0,00	3,83	7,94		0,00	3,83	17,34		0,00	5,23	20,62
10,75	12,50	8,25		10,75	12,50	9,86		10,75	10,03	12,35
36,57	35,93	29,15		36,57	35,93	19,23		36,57	32,54	15,98
3,83	0,00	6,80		3,83	0,00	16,91		3,83	3,40	20,28
7,94	6,80	0,00		7,94	6,80	10,11		7,94	3,40	13,49
M1-3	M2-3	M3-3		M1-4	M2-4	M3-4		M1-5	M2-5	M3-5
16,38	10,63	10,40		15,69	9,06	10,40		15,69	6,04	20,81
0,00	5,86	26,35		1,91	8,50	26,35		1,91	10,93	36,57
10,75	6,69	18,53		11,50	4,12	18,53		11,50	5,90	27,89
36,57	30,71	10,40		36,20	28,23	10,40		36,20	25,64	0,00
3,83	6,12	25,55		1,91	9,18	25,55		1,91	10,73	35,93
7,94	2,83	18,79		7,14	4,12	18,79		7,14	4,18	29,15



# Task 7

- We assume that the data warehouse of a company contains an employee dimension and that all the information on employees is stored in an employee table. The following table contains aggregated information for the employees. The last column (attribute amount) covers the amount of entries in the employee table for the combinations of the attribute values shown in the other columns.

- Create a decision tree (2 split levels) with the target attribute salary. Use the gini index as split criteria with the following generalization: If a set  $T$  contains elements from  $n$  different classes, then following formula is valid:

$$gini(T) = 1 - \sum_{i=1}^n p_i^2$$

Department	Status	age	salary	amount
sale	manager	31 ... 35	46K ... 50K	30
Sale	Employee	26 ... 30	26K ... 30K	40
Sale	Employee	31 ... 35	31K ... 35K	40
Infrastructure	Employee	21 ... 25	46K ... 50K	20
Infrastructure	manager	31 ... 35	66K ... 70K	5
Infrastructure	Employee	26 ... 30	46K ... 50K	3
Infrastructure	manager	41 ... 45	66K ... 70K	3
Marketing	manager	36 ... 40	46K ... 50K	10
Marketing	Employee	31 ... 35	41K ... 45K	4
Office	manager	46 ... 50	36K ... 40K	4
Office	Employee	26 ... 30	26K ... 30K	6

- How many wrong classifications do you get if the training data is classified by means of the created decision tree?

# Decision Tree Induction: Generic Algorithm

```
Node generateDecisionTree(samples, attributeList) begin
  create a new node n;
  if samples are all of class c then begin
    label leaf node n with c;
    return n;
  else if attributeList ==  $\emptyset$  then begin
    label leaf node n with majorityClass(n);
    return n;
  end;
  select bestSplit from attributeList; // Depends on specific algorithm!

  for each value v of attribute bestSplit do begin
    sv = set of samples with (bestSplit == v);
    if sv ==  $\emptyset$  then begin
      create a new node nv and label it with majorityClass(samples)
    else
      nv = generateDecisionTree(sv, attributeList \ bestSplit);
    end;
    create a branch from n to nv labeled with the test (bestSplit == v);
  end;
  return n;
end;
```

# Example Split Criterion: Gini Index of CART-Algorithm

Given:

- two classes: Pos und Neg
- a data set  $S$  with  $p$  elements in class Pos and  $n$  elements in Neg

$$f_p = p / (p + n)$$

$$f_n = n / (p + n)$$

$$\text{gini}(S) = 1 - f_p^2 - f_n^2$$

Let  $S_1$  and  $S_2$  be data sets, where  $S_1 \cup S_2 = S$  and  $S_1 \cap S_2 = \emptyset$

$$\text{gini}_{\text{split}}(S_1, S_2) = \text{gini}(S_1) \times (p_1 + n_1) / (p + n) + \text{gini}(S_2) \times (p_2 + n_2) / (p + n)$$

**Aim:** Find  $S_1$  and  $S_2$  such that  $\text{gini}_{\text{split}}(S_1, S_2)$  is minimized.

# Splits on age

department	Status	age	salary	qty
Sales	manager	31 ... 35	46K ... 50K	30
Sales	employee	26 ... 30	26K ... 30K	40
Sales	employee	31 ... 35	31K ... 35K	40
infrastructure	employee	21 ... 25	46K ... 50K	20
infrastructure	manager	31 ... 35	66K ... 70K	5
infrastructure	employee	26 ... 30	46K ... 50K	3
infrastructure	manager	41 ... 45	66K ... 70K	3
Marketing	manager	36 ... 40	46K ... 50K	10
Marketing	employee	31 ... 35	41K ... 45K	4
office	manager	46 ... 50	36K ... 40K	4
office	employee	26 ... 30	26K ... 30K	6

↓

$$\text{Gini}(21-25) = 1 - (20/20)^2 = 0$$

↓ ↓ ↓ ↓ ↓ ↓

$$\text{Gini}(26-50) = 1 - (43/145)^2 - (46/145)^2 - (40/145)^2 - (8/145)^2 - (4/145)^2 - (4/145)^2 = 0,731$$

$$\text{Gini\_split} = \text{gini}(A) * (20/165) + \text{gini}(B) * (145/165) = 0,642$$

A	B	Gini(A)	Gini(B)	gini_split
21 - 25	26 - 50	0	0,731	0,642
21 - 30	31 - 50	0,444	0,642	0,56
21 - 35	36 - 50	0,700	0,567	0,687
21 - 40	41 - 50	0,691	0,490	0,682
21 - 45	46 - 50	0,700	0	0,683

Best 1. Split: age < 31

Other possible splits: Dep., status

## 2nd level: Splits on department, age < 31

department	Status	age	salary	qty
Sales	manager	31 ... 35	46K ... 50K	30
Sales	employee	26 ... 30	26K ... 30K	40
Sales	employee	31 ... 35	31K ... 35K	40
infrastructure	employee	21 ... 25	46K ... 50K	20
infrastructure	manager	31 ... 35	66K ... 70K	5
infrastructure	employee	26 ... 30	46K ... 50K	3
infrastructure	manager	41 ... 45	66K ... 70K	3
Marketing	manager	36 ... 40	46K ... 50K	10
Marketing	employee	31 ... 35	41K ... 45K	4
office	manager	46 ... 50	36K ... 40K	4
office	employee	26 ... 30	26K ... 30K	6

↓

$$\text{Gini}(\text{Sales}) = 1 - (40/40)^2 = 0$$

↓      ↓

$$\text{Gini}(\text{I}, \text{M}, \text{O}) = 1 - (23/29)^2 - (6/29)^2 = 0,328$$

$$\begin{aligned} \text{Gini\_split} &= \text{gini}(\text{A}) * (40/69) + \text{gini}(\text{B}) * (29/69) \\ &= 0,138 \end{aligned}$$

A	B	Gini(A)	Gini(B)	gini_split
Sales	infrastructure, Marketing, office	0	0,328	0,138
infrastructure	Sales, Marketing, office	0	0	0
Marketing	Sales, infrastructure, office	0	0,444	0,444
office	Sales, infrastructure, Marketing	0	0,464	0,423
Sales, infrastructure	Marketing, office	0,464	0	0,423
Sales, Marketing	infrastructure, office	0	0,328	0,138
Sales, office	infrastructure, Marketing	0	0	0

Best 2. Split: department = {Sales, office}

## 2. level: Splits on Status, **age** $\geq 31$

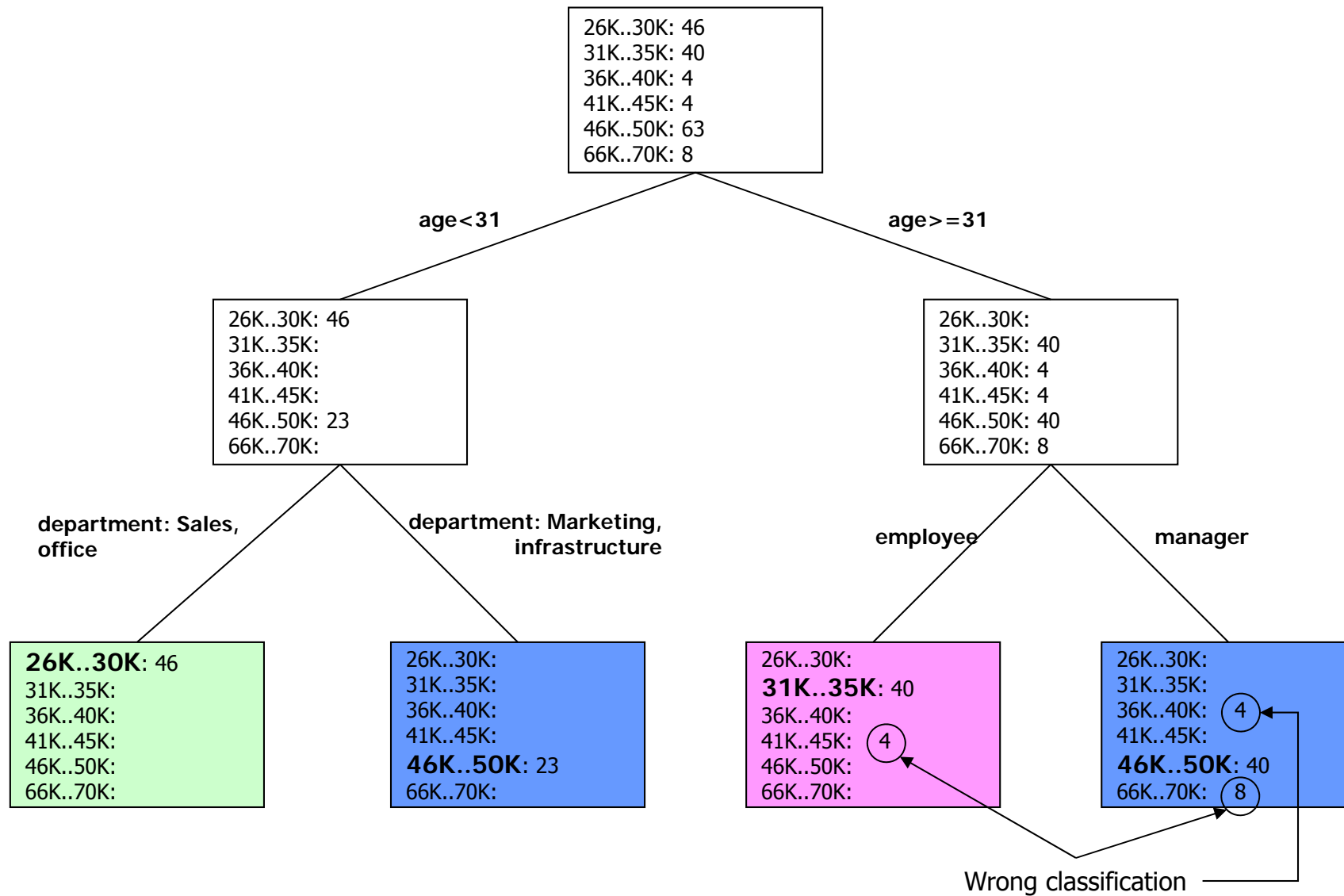
department	Status	age	salary	qty
Sales	manager	31 ... 35	46K ... 50K	30
Sales	employee	26 ... 30	26K ... 30K	40
Sales	employee	31 ... 35	31K ... 35K	40
infrastructure	employee	21 ... 25	46K ... 50K	20
infrastructure	manager	31 ... 35	66K ... 70K	5
infrastructure	employee	26 ... 30	46K ... 50K	3
infrastructure	manager	41 ... 45	66K ... 70K	3
Marketing	manager	36 ... 40	46K ... 50K	10
Marketing	employee	31 ... 35	41K ... 45K	4
office	manager	46 ... 50	36K ... 40K	4
office	employee	26 ... 30	26K ... 30K	6

$$\text{Gini(manager)} = 1 - \left(\frac{4}{52}\right)^2 - \left(\frac{40}{52}\right)^2 - \left(\frac{8}{52}\right)^2 = 0,379$$

$$\text{Gini(employee)} = 1 - \left(\frac{40}{44}\right)^2 - \left(\frac{4}{44}\right)^2 = 0,165$$

$$\text{Gini\_split} = \text{gini(manager)} * (52/96) + \text{gini(employee)} * (44/96) = 0,281$$

**Best 2. Split: Status = manager**





# Exercise 7

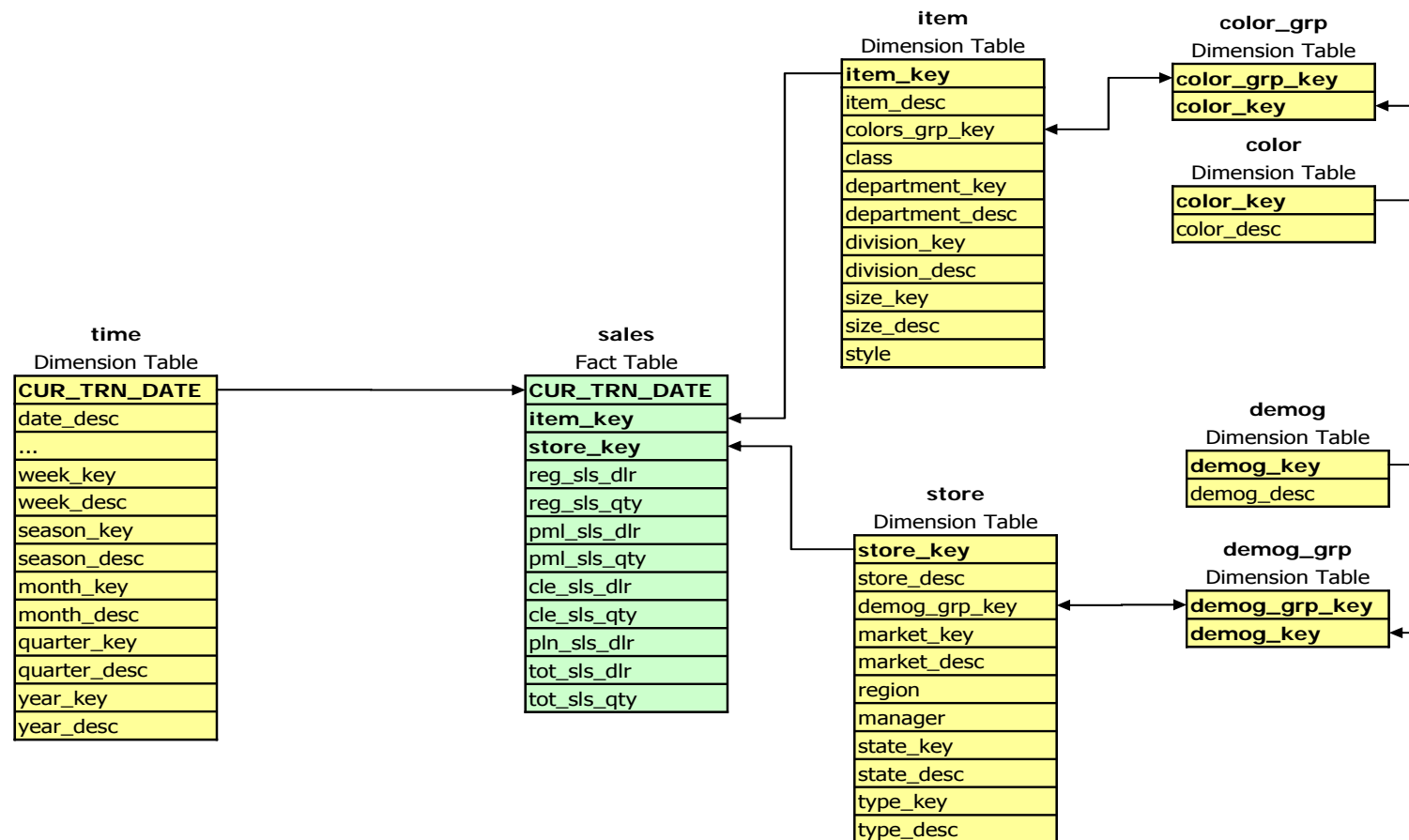
# Task 1

- Consider the star schema from Task 5.4.
- 1. Which options do you have to provide materialized summary data for the fact table sales?
- 2. Indicate one materialized view that can be used for query 1 of Task 5.4 and has a low runtime for this query?
- 3. Use the algorithms introduced in the lecture to determine a set of materialized views for the grouping options indicated below. The available storage is sufficient for 5400 rows of materialized views. Use the number of rows in each of these materialized views as a measure for the query costs when querying a view. Start by building the dependency graph.

grouping	Number of rows
C, I, S	5000
Y, I, S	3500
C, I	600
Y, I	600
I, S	300
C, S	185
Y, S	185
S	180
I	120
Y	50
C	50
()	1

C = CUR\_TRN\_DATE  
 I = item\_key  
 S = store\_key  
 Y = year\_key

# Star Schema

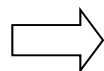


**sales**

Fact Table

<b>CUR_TRN_DATE</b>
<b>item_key</b>
<b>store_key</b>
reg_sls_dlr
reg_sls_qty
pml_sls_dlr
pml_sls_qty
cle_sls_dlr
cle_sls_qty
pln_sls_dlr
tot_sls_dlr
tot_sls_qty

- Views of different groupings (1 Attr., 2 Attr.)
- For each fact attribute itself or in combination
- with aggregate functions
- With combinations consisting of more facts together



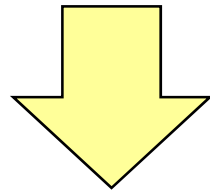
unlimited set of views

# Query 1

```
SELECT  ti.year_key AS year, ti.month_key AS month,
        it.department_key, st.store_key,
        SUM(reg_sls_dlr) AS reg_sls_dlr,
        SUM(reg_sls_qty) AS reg_sls_qty,
        SUM(tot_sls_dlr) AS tot_sls_dlr,
        SUM(pln_sls_dlr) AS pln_sls_dlr
FROM      DW2.sales sa, DW2.time ti, DW2.item it, DW2.store st
WHERE     ti.cur_trn_date = sa.cur_trn_date
AND       sa.item_key      = it.item_key
AND       sa.store_key     = st.store_key
GROUP BY
ROLLUP(
    ti.year_key, it.department_key, st.store_key, ti.month_key);
```

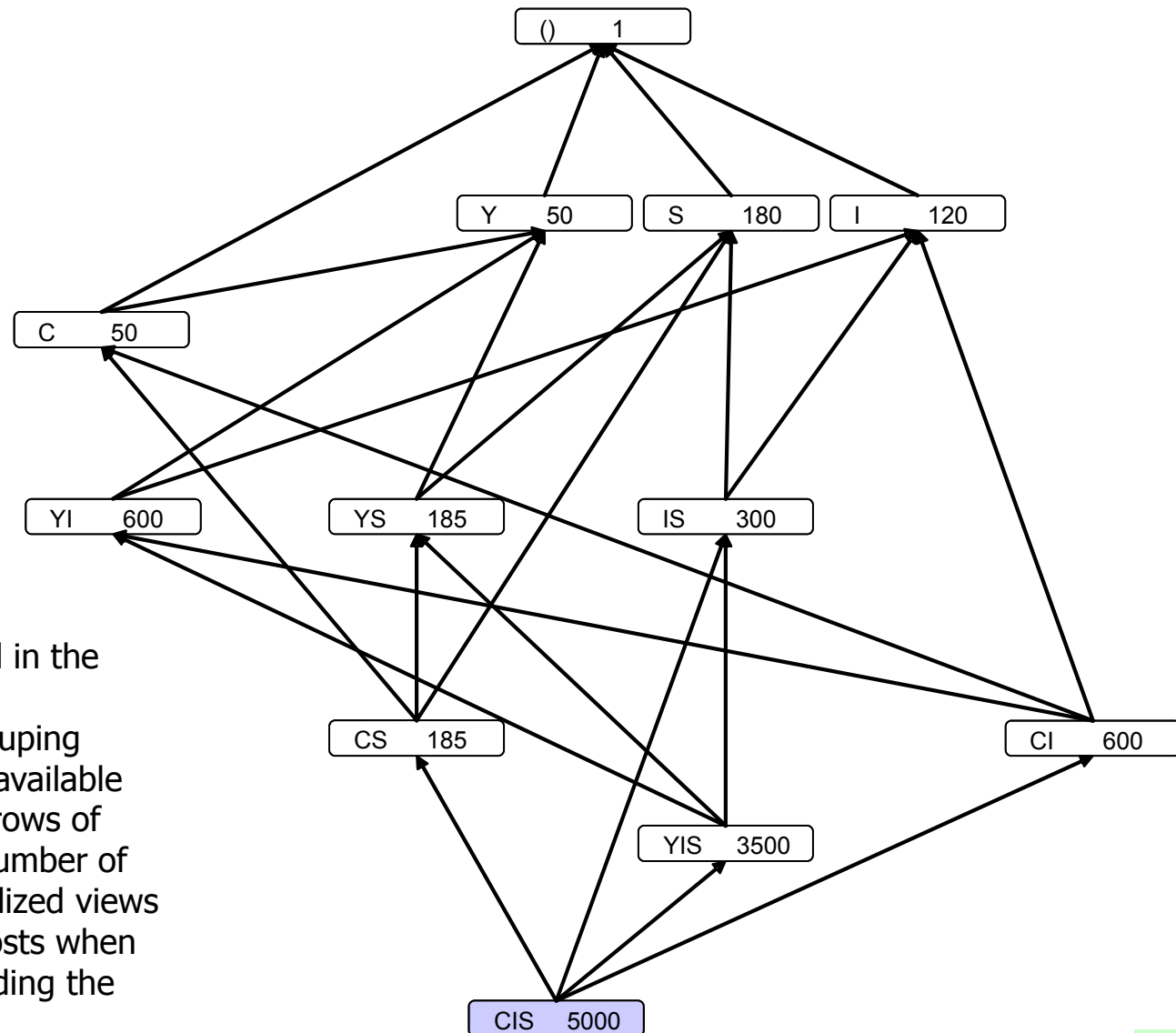
# Example for Materialized View

```
SELECT  ti.year_key AS year, ti.month_key AS month,
        it.department_key, st.store_key,
        SUM(reg_sls_dlr) AS reg_sls_dlr,
        SUM(reg_sls_qty) AS reg_sls_qty,
        SUM(tot_sls_dlr) AS tot_sls_dlr,
        SUM(pln_sls_dlr) AS pln_sls_dlr
FROM    DW2.sales sa, DW2.time ti, DW2.item it, DW2.store st
WHERE   ti.cur_trn_date = sa.cur_trn_date
AND     sa.item_key      = it.item_key
AND     sa.store_key     = st.store_key
GROUP BY ti.year_key, it.department_key, st.store_key, ti.month_key
```



store result as materialized table/view

supports various grouping sets



Use the algorithms introduced in the lecture to determine a set of materialized views for the grouping options indicated below. The available storage is sufficient for 5400 rows of materialized views. Use the number of rows in each of these materialized views as a measure for the query costs when querying a view. Start by building the dependency graph.

## Algorithm per size

grouping	Amount of tuples
Y, I, S	3500
C, I	600
Y, I	600
I, S	300
C, S	185
Y, S	185
S	180
I	120
Y	50
C	50
( )	1

$M=(CIS) \quad S=400$

$M=(CIS, ()) \quad S=399$

$M=(CIS,(),C) \quad S=349$

$M=(CIS,(),C,Y) \quad S=299$

$M=(CIS,(),C,Y,I)$



# View Selection Algorithm

A: Set of all possible materialized views

S: Space for materialized views

M: Set of selected materialized views

```

select_views(A, S) begin
    M = {most granular materialized view}           // adjust available space S
    while S > 0 do begin
         $u_{opt} = \emptyset$ ;
        BPUS( $\{u_{opt}\}, M$ ) = 0;
        for each  $u \in (A - M)$  do begin
            B( $\{u\}, M$ ) = 0;
            for each  $q \in \text{descendants}(u)$  do
                B( $\{u\}, M$ ) = B( $\{u\}, M$ ) + C( $q, M$ ) - C( $q, M \cup \{u\}$ )
            BPUS( $\{u\}, M$ ) = B( $\{u\}, M$ ) / u.size
            if BPUS( $\{u\}, M$ ) > BPUS( $\{u_{opt}\}, M$ ) then
                 $u_{opt} = u$ ;
        end;
        if S -  $u_{opt}.size$  > 0 then
            M = M  $\cup \{u_{opt}\}$ ;
            S = S -  $u_{opt}.size$ ;
        else
            S = 0;
        end;
    end;
    return M;
end;

```

# Task 1

u	cost	amount	savings $B(\{u\}, M)$	ratio $BPUS(\{u\}, M)$
(YIS)	3500	8	12000	3,43
(YS)	185	4	19260	104,11
(IS)	300	4	18800	62,67
(CI)	600	6	26400	44
(YI)	600	4	17600	29,33
(CS)	185	6	28890	156,16
(Y)	50	2	9900	198
(I)	120	2	9760	81,33
(C)	50	3	14850	297
(S)	180	2	9640	53,56
( )	1	1	4999	4999

$$S = 5400 - 5000 = 400$$

$$M = \{(CIS)\}$$

$$B(\{u\}, M) = (5000 - 3500) * 8$$

$$BPUS = 12000 / 3500$$

$$S = 400 - 1 = 399$$

$$M = \{(CIS), ( )\}$$

# Task 1

u	cost	amount	savings $B(\{u\}, M)$	ratio $BPUS(\{u\}, M)$
(YIS)	3500	7	10500	3
(YS)	185	3	14445	78,08
(IS)	300	3	14100	47
(CI)	600	5	22000	36,67
(YI)	600	3	13200	22
(CS)	185	5	24075	130,14
(Y)	50	1	4950	99
(I)	120	1	4880	40,67
(C)	50	2	9900	198
(S)	180	1	4820	26,78

$$S = 399 - 50 = 349$$

$$M = \{(CIS), (), (C)\}$$

# Task 1

u	cost	amount	savings $B(\{u\}, M)$	ratio $BPUS(\{u\}, M)$
(YIS)	3500	7	10500	3
(YS)	185	3	14445	78,08
(IS)	300	3	14100	47
(CI)	600	4	17600	29,33
(YI)	600	3	13200	22
(CS)	185	4	19260	104,11
(Y)	50	1	0	0
(I)	120	1	4880	40,67
(S)	180	1	4820	26,78

$$S = 349 - 185 = 164$$

$$M = \{(CIS), (), (C), (CS)\}$$

# Task 1

u	cost	amount	savings $B(\{u\}, M)$	ratio $BPUS(\{u\}, M)$
(YIS)	3500	7	10500	3
(YS)	185	3	0	0
(IS)	300	3	14100	47
(CI)	600	4	17600	29,33
(YI)	600	3	13200	22
(Y)	50	1	0	0
(I)	120	1	4880	40,67
(S)	180	1	5	0,03

$$S = 164 - 185 < 0!$$

$$M = \{(\{(CIS), (), (C), (CS)\})\}$$

$$S = 164 - 120 = 44$$

$$M = \{(\{(CIS), (), (C), (CS), (I)\})\}$$

## Task 2

1. In a bitmap index the TID lists in the leaves of a B-tree are replaced by bit lists. Determine the important factors that define the needed storage. Under which conditions do bit lists need less storage than TID lists?
2. For the processing of some queries several bit lists have to be combined. Indicate the calculation of the predicate A IN (3,4,5,6) by means of the following bit list for attribute A:

**A:**

1: B1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0
2: B2	1	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
3: B3	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
4: B4	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0
5: B5	0	0	0	0	1	0	0	0	1	1	0	1	0	0	0	0
6: B6	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
7: B7	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
8: B8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

- In a different encoding of bitmap indexes bit lists are deduced from the bitmaps  $B_j$  as follows:
- $C_j[i] = 1$ , if  $B_j[i]=1$  or  $B_k[i]=1$  for  $k < j$
- Determine the bitmap index for attribute A according to this encoding and use it to calculate predicate A IN (3,4,5,6).

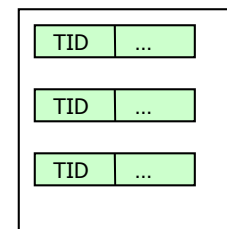
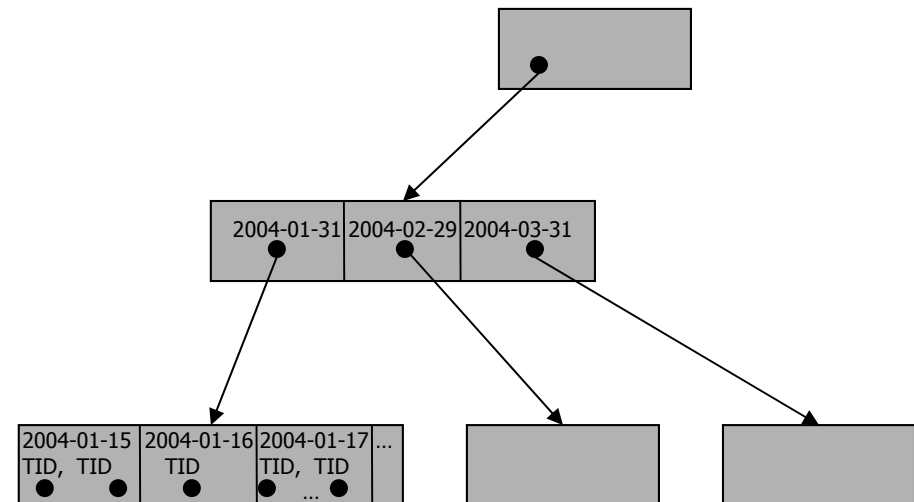
## Task 2

$n$ : number of rows in table  
 $L_{TID}$ : number of bytes to store TID  
 $k$ : number of key values  
 $L_{key}$ : number of bytes to store key

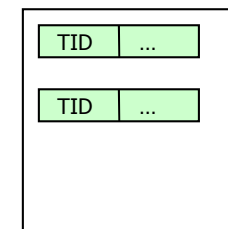
$L1$ : storage needed for TIDs [in bit]

$$L1 = k * L_{key} * 8 + n * L_{TID} * 8$$

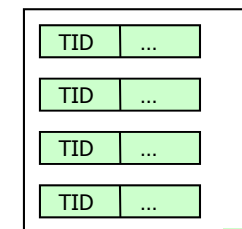
B-Tree on  
attribute date



block 324



block 325



block 326

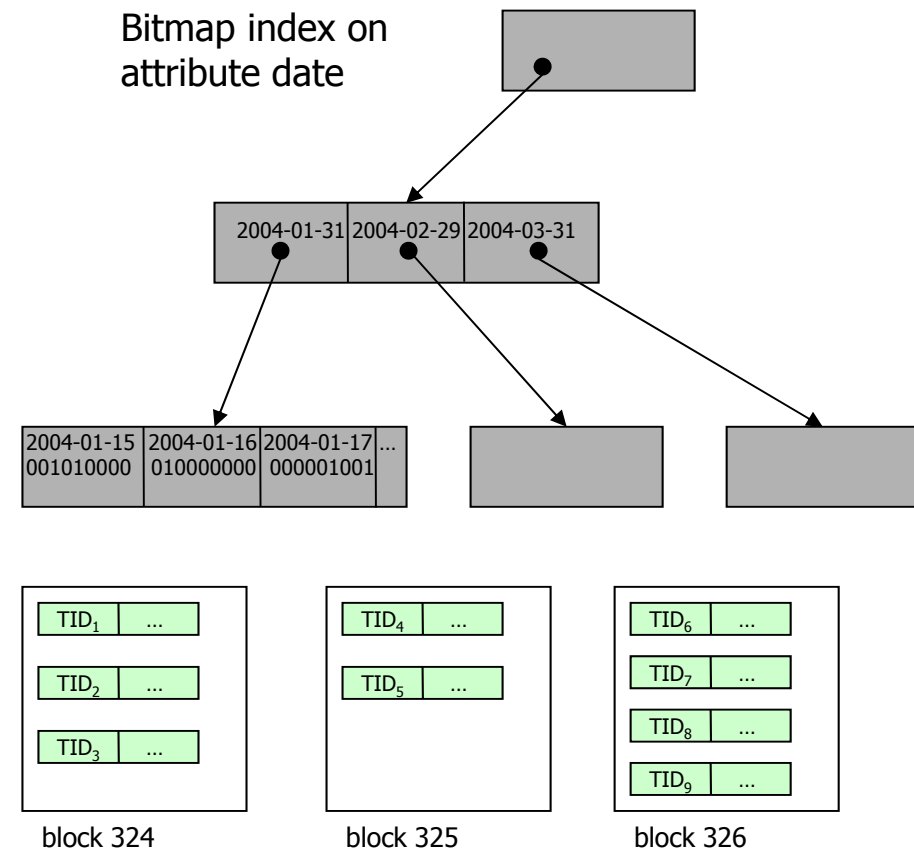
## Task 2

$n$ : number of rows in table  
 $k$ : number of key values  
 $L_{\text{key}}$ : number of bytes to store key

L2: storage needed for bit lists [in bit]

$$L2 = k * L_{\text{key}} * 8 + k * n$$

2004-01-15: 001010000  
 2004-01-16: 010000000  
 2004-01-17: 000001001





## Task 2

### comparision of storage space

$L2 < L1 :$

$$k*n + k*L_{key}*8 < n*L_{TID}*8 + k*L_{key}*8$$

$$k*n < n*L_{TID}*8$$

$$k < L_{TID} * 8$$

$L_{TID}$ : number of bytes to store TID

$k$ : number of key values

## Task 2

A:

1: B1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0
2: B2	1	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
3: B3	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
4: B4	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0
5: B5	0	0	0	0	1	0	0	0	1	1	0	1	0	0	0	0
6: B6	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
7: B7	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
8: B8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

B3[i] or B4[i] or B5[i] or B6[i]:  
 0 0 1 1 1 0 0 0 1 1 1 1 1 0 0 0

A:

1: C1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0
2: C2	1	1	0	0	0	1	1	1	0	0	0	0	0	0	1	0
3: C3	1	1	1	0	0	1	1	1	0	0	0	0	0	0	1	0
4: C4	1	1	1	1	0	1	1	1	0	0	1	0	0	0	1	0
5: C5	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	0
6: C6	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
7: C7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
8: C8	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

C6[i] and not C2[i]:

0 0 1 1 1 0 0 0 1 1 1 1 1 0 0 1 0