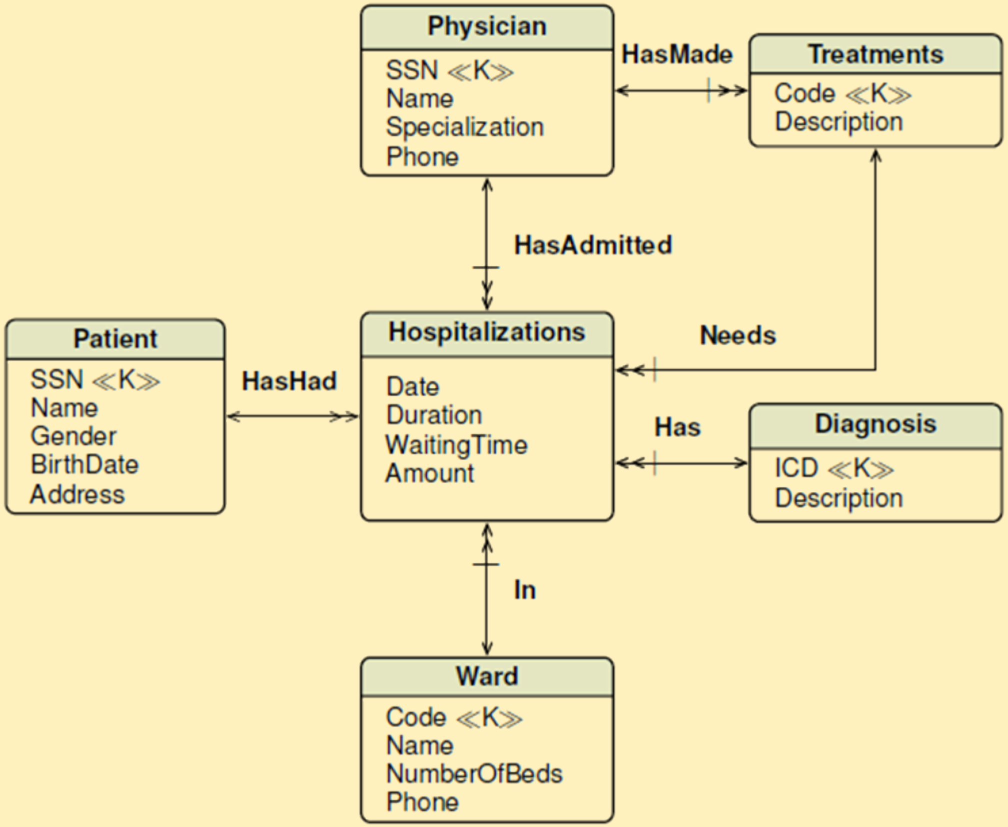
OPEN LAB: HOSPITAL

An hospital needs a DM to extract information from their operational database with information about inpatients treatments.



1. Total billed amount for hospitalizations, **by** diagnosis code and description, **by** month (year).
2. Total number of hospitalizations and billed amount, **by** ward, **by** patient gender (age at date of admission, city, region).
3. Total billed amount, average length of stay and average waiting time, **by** diagnosis code and description, **by** name (specialization) of the physician who has admitted the patient.
4. Total billed amount, and average waiting time of admission, **by** patient age (region), **by** treatment code (description).

REQUIREMENTS SPECIFICATION

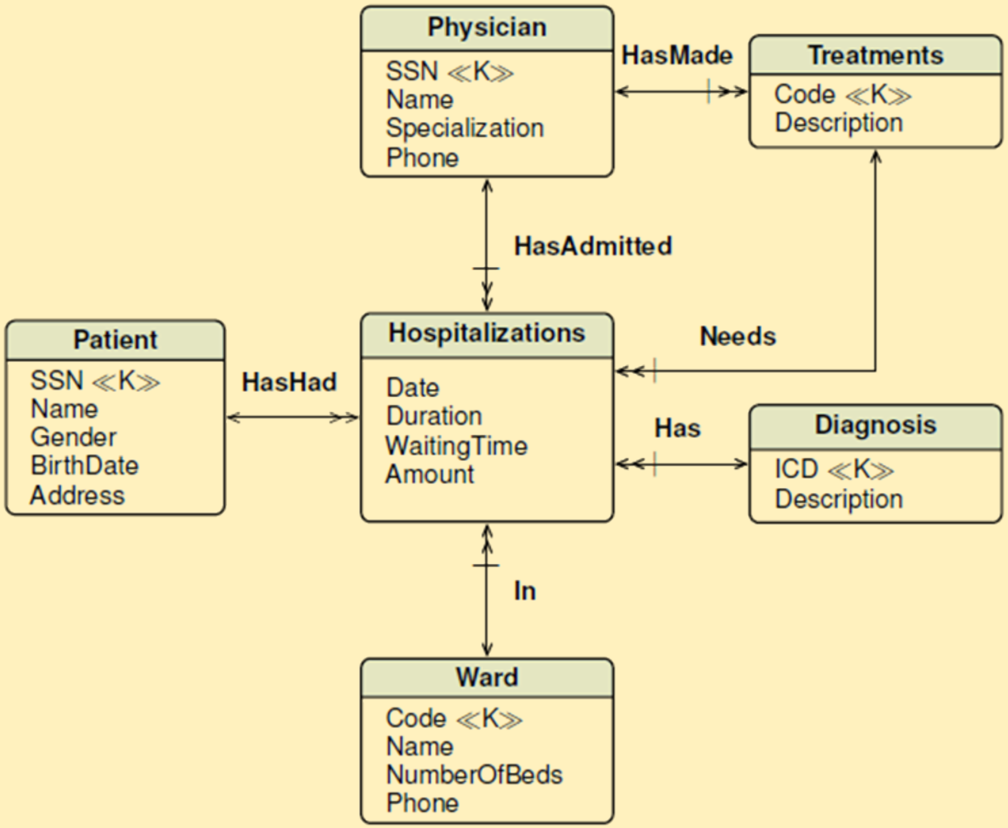
Hospitalization

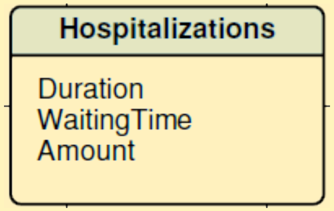
Requirements analysis	Dimensions	Measures	Metrics

REQUIREMENTS SPECIFICATION

	Fact granularity
Description	
Preliminary dimensions	
Preliminary measures	

HOSPITALIZATIONS DATA MART CONCEPTUAL SCHEMA





DATA BASE

DATA MART

SUMMARY

The **analysis-driven** design of a data mart.

Business questions

For a data subsets to use,

the metrics to compute,

grouping data **by dimensions (attributes)**,

how the result should be presented.

SELECT X **FROM** … **WHERE** B **GROUP BY** Y **ORDER BY** W

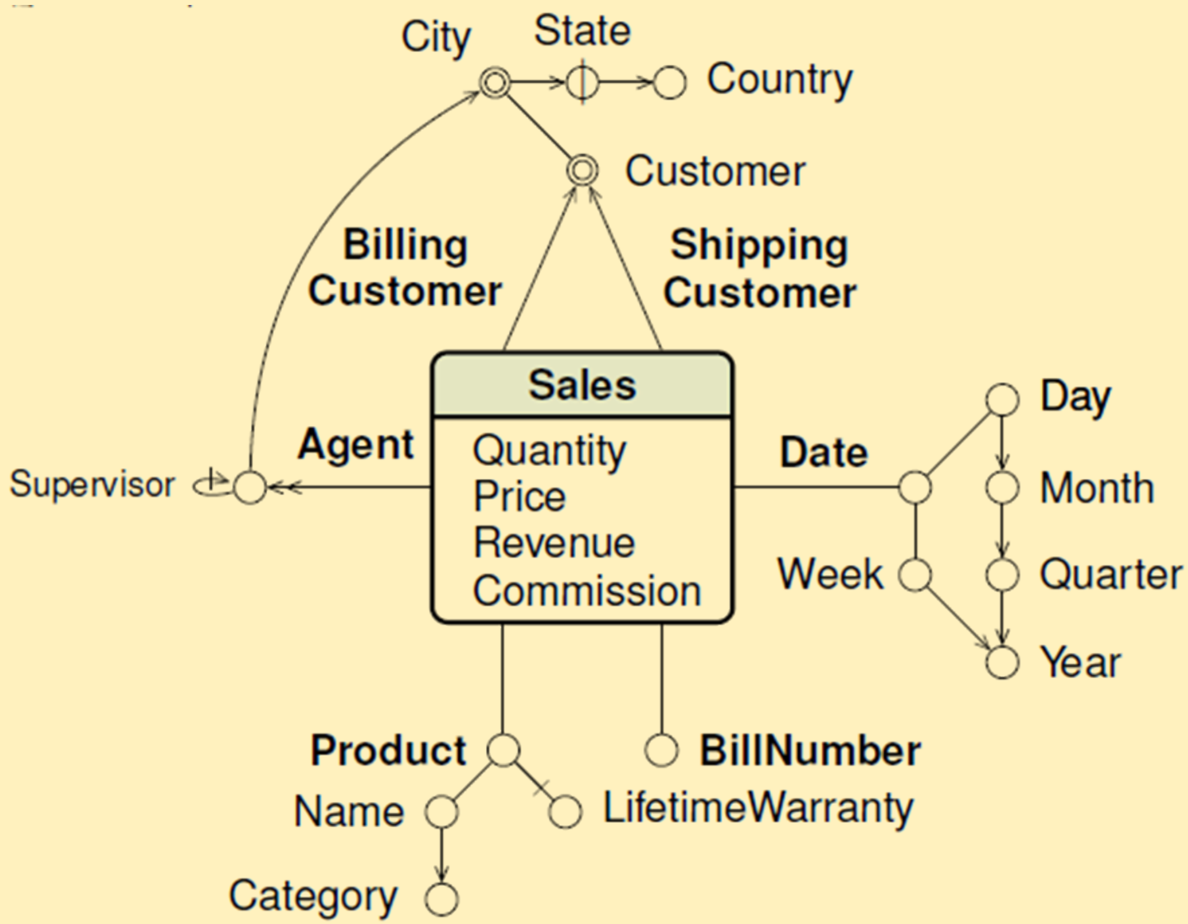
Alternative: Types of reports to be produced

Facts granularity, measures and their types, dimensions

Data availability

MORE ABOUT DATA MART CONCEPTUAL MODELLING

- Degenerate dimensions
- Facts descriptive attributes
- Optional dimensions or attributes
- Multivalued dimensions
- Hierarchies types
- Shared hierarchies



RELATIONAL MODEL

Relational OLAP systems are relational DBMS extended with specific features to support business intelligence analysis.

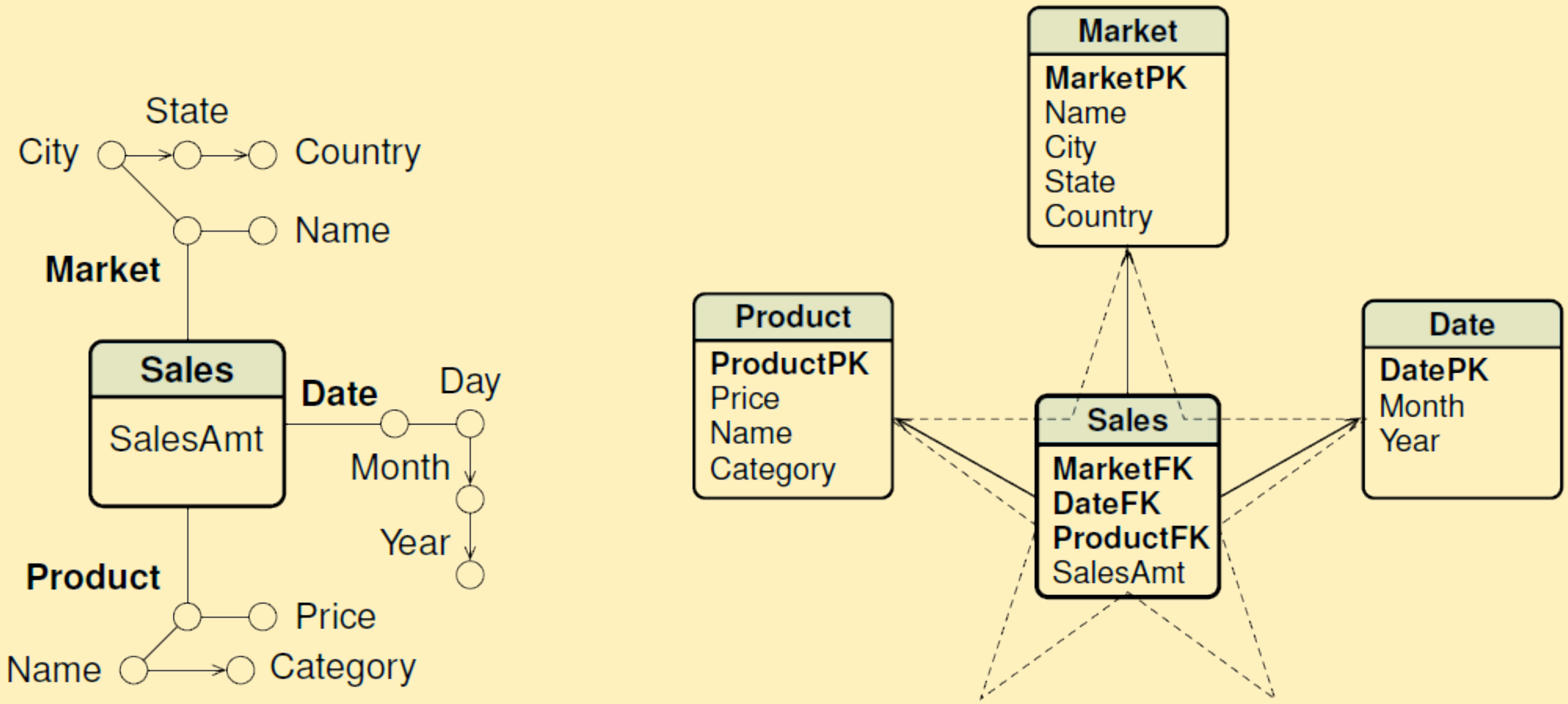
A DW is represented with a special kind of **relational schema**

A **star schema**,

A **snowflake schema** or

A **constellation schema**.

A STAR SCHEMA EXAMPLE

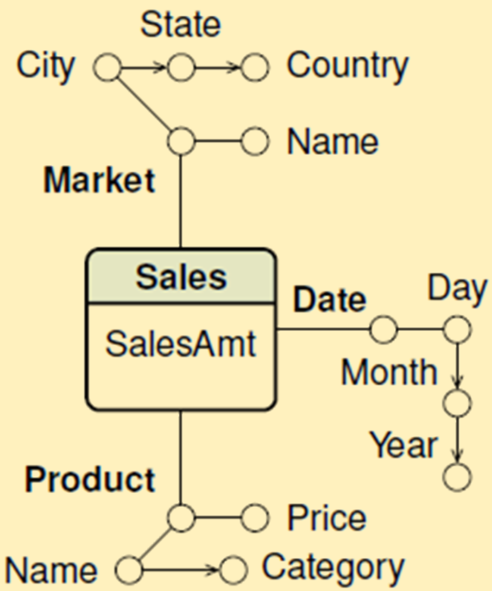


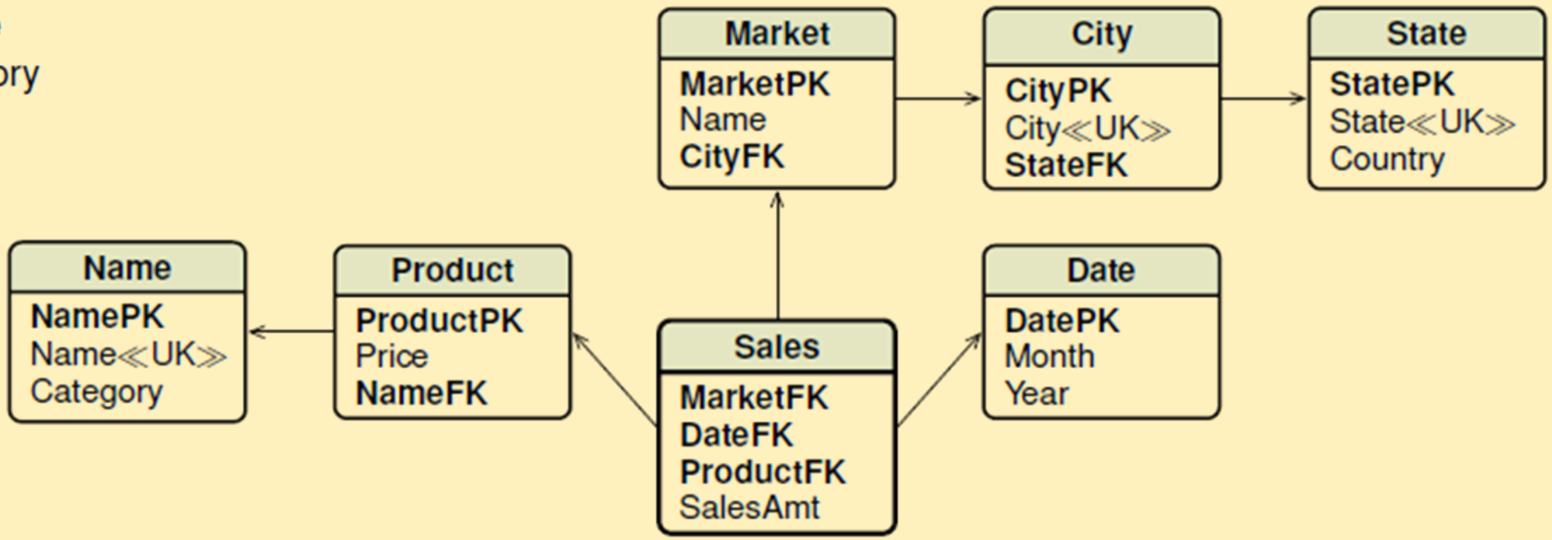
(a) *The conceptual design* (b) *The relational design*

In a data mart relational schema a **dimension table always uses a system-generated primary key**, called a **Surrogate Key, to support Type 2 technique of slowly changing dimensions.**

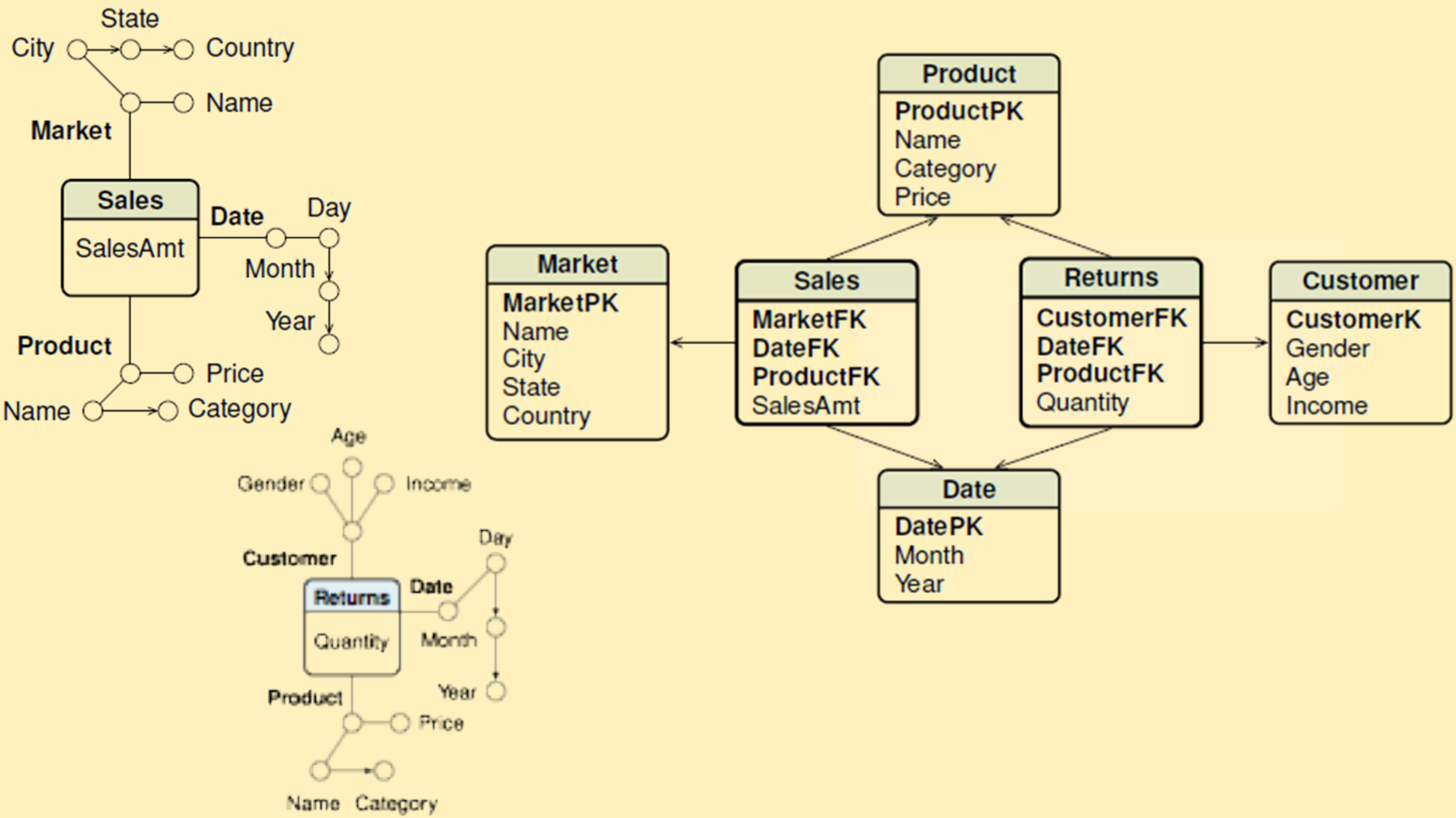
And the fact table key?

SNOWFLAKE SCHEMA



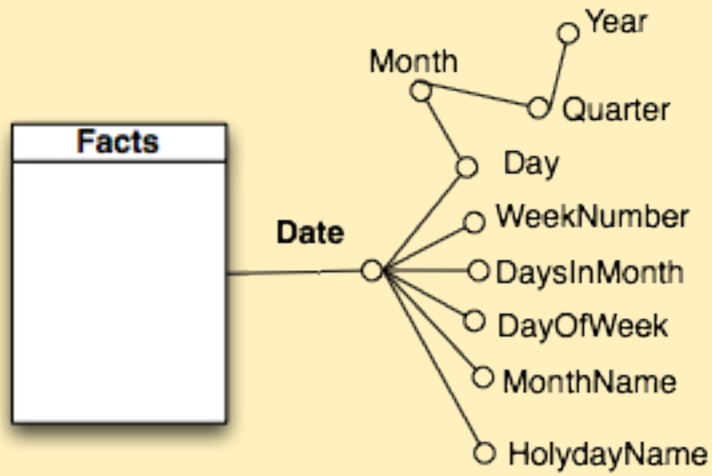


CONSTELLATION SCHEMA



Università di Pisa

Hyp: **Date** at daily grain



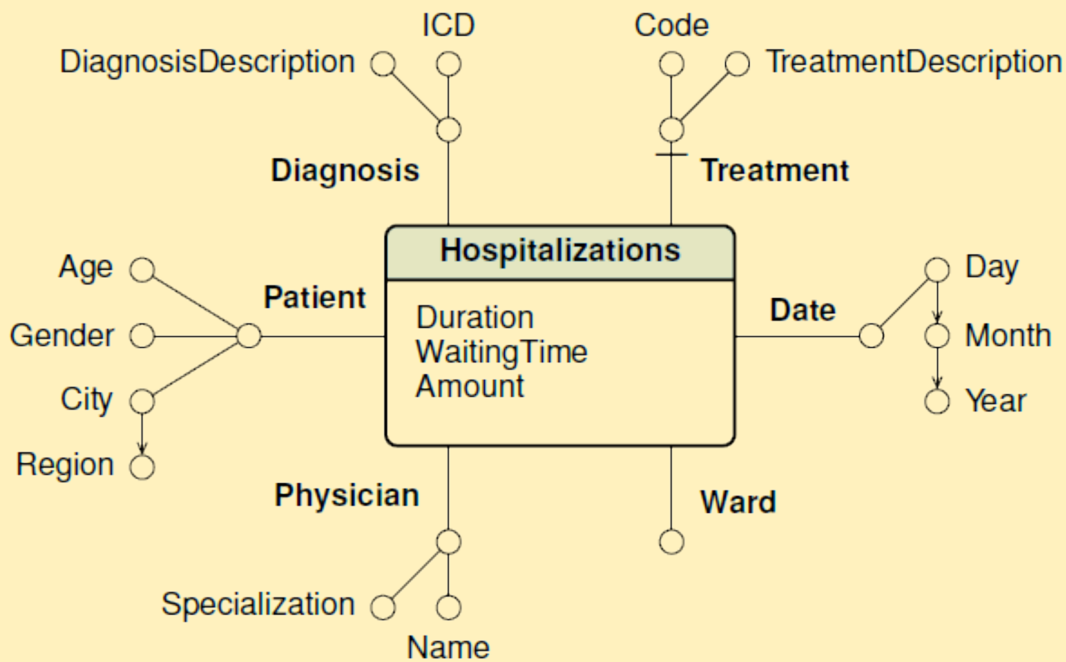
In the logical schema,
the dimension **Date** has the surrogate key with the integer value
YYYYMMDD

DATE

Attribute Name	Type	Format/Example
DatePK	int	YYYYMMDD
Month	int	YYYYMM
Quarter	int	YYYYQ
Year	int	YYYY
WeekNumber	int	1 to 52 or 53
DayInMonth	int	1 to 31
DayOfWeek	string	Monday
MonthName	string	January
HolydayName	string	Easter

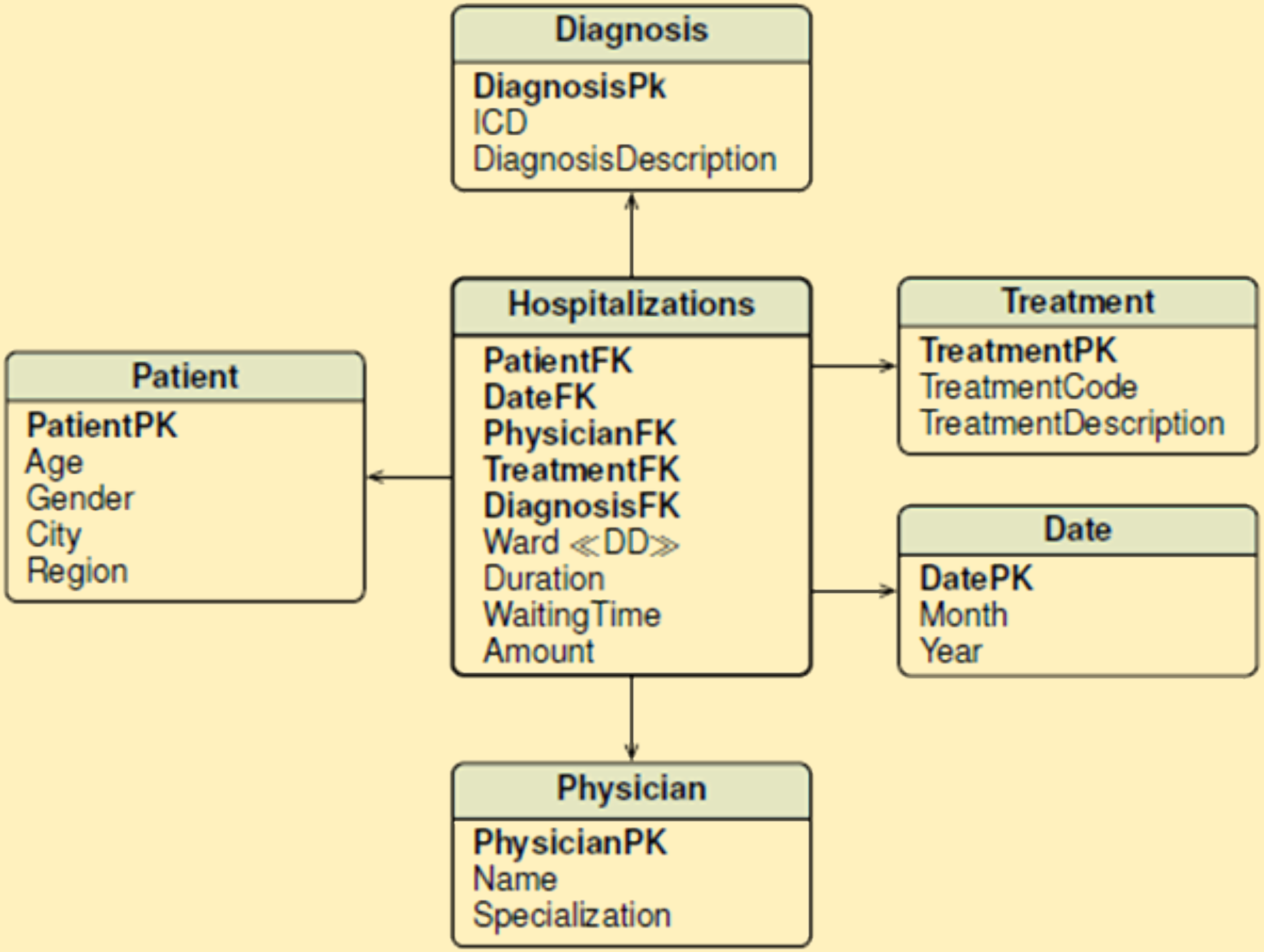
HOSPITALIZATIONS DATA MART CONCEPTUAL SCHEMA

UNIVERSITÀ DI PISA



DESIGN THE LOGICAL SCHEMA

HOSPITALIZATIONS: INITIAL LOGICAL SCHEMA



AIRLINE COMPANIES: REQUIREMENTS SPECIFICATION

Airline companies

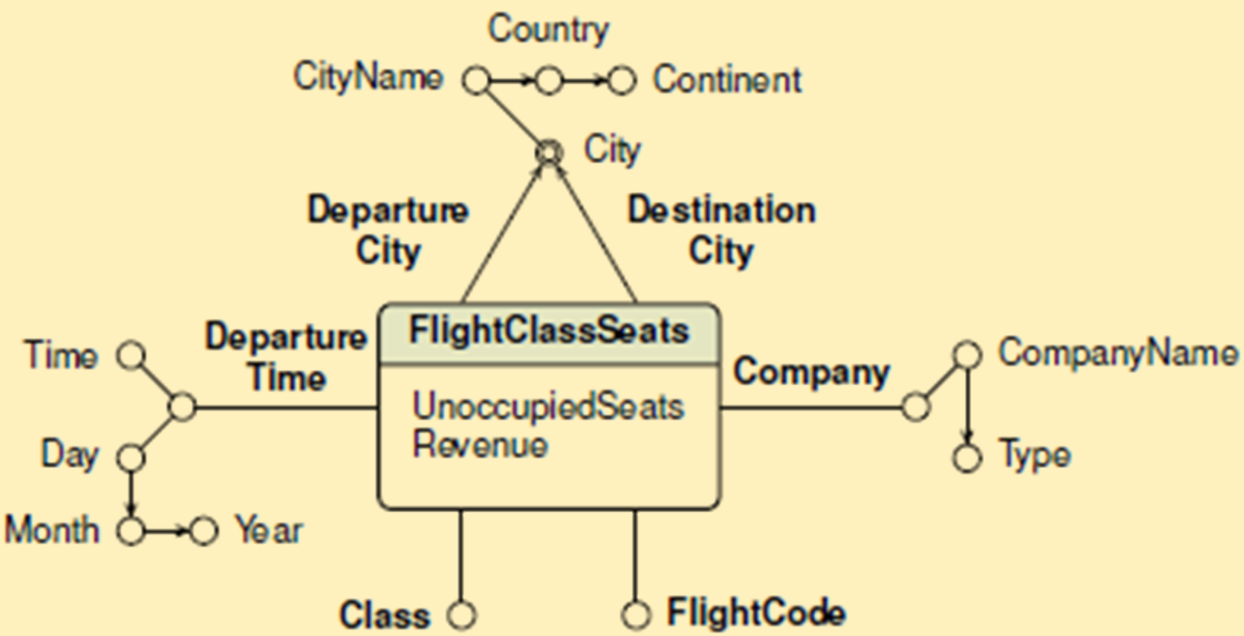
Requirements analysis	Dimensions	Measure	Metrics
Number of unoccupied seats in a given year, by flight code, by company name (or type), by class, by departure time (time, day, month, year)	FlightCode, Class, Company(Name, Type), DepartureTime (Time, Day, Month, Year)	UnoccupiedSeats	Total UnoccupiedSeats
Number of unoccupied seats in a given class and year, by flight code, by company name, by class, by departure (destination) city (country, continent).	FlightCode, Class, Company(Name), DepartureCity (Country, Continent), DestinationCity (Country, Continent)	UnoccupiedSeats	Total UnoccupiedSeats
Number of unoccupied seats and revenue of the Alitalia company, by year, by month, by destination country.	Company(Name), DepartureTime (Month, Year), DepartureCity(Country)	UnoccupiedSeats Revenue	Total UnoccupiedSeats, Revenue

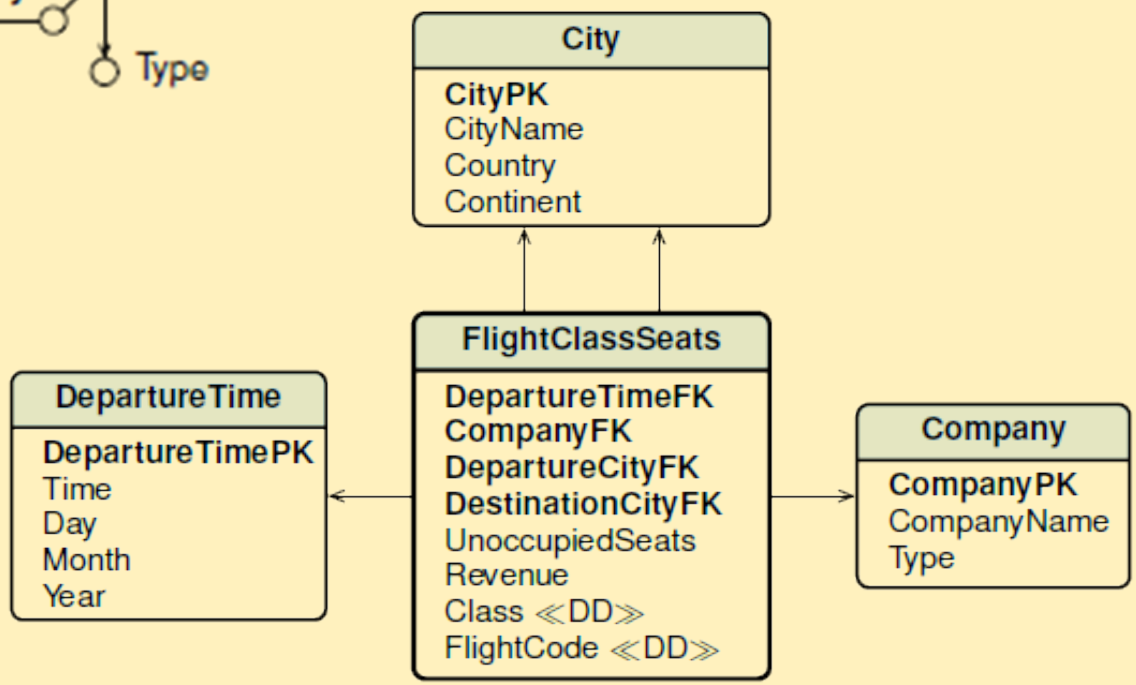
Fact granularity

Description	A fact is the information on the number of unoccupied seats on a flight of a class of a company
Preliminary dimensions	Class, FlightCode, Company, Departure time, Departure city, Destination city
Preliminary measures	UnoccupiedSeats, Revenue

AIRLINE COMPANIES: CONCEPTUAL AND LOGICAL DESIGN

Università di Pisa





A dimensional attributes hierarchy models **attributes dependency**, i.e. a **functional dependency** between attributes, using the relational model terminology.

Definition 8.1 *Functional Dependency*

Given a relation schema R and X, Y subsets of attributes of R, a functional dependency $X \rightarrow Y$ (X *determines* Y) is a constraint that specifies that for every possible instance r of R and for any two tuples $t_1, t_2 \in r$, $t_1[X] = t_2[X]$ implies $t_1[Y] = t_2[Y]$.

For example, the dimension **Date** has attributes **Month**, **Quarter**, **Year**. Can we define a **dimensional hierarchy** among them?

Month → Quarter → Year

PkDate → Month, Quarter, Year
Month → Quarter
Quarter → Year

Attention to the attribute values !

Date **Month → Quarter → Year**

PkDate	Month	Quarter	Year
20080101	200801	20081	2008
20080102	200801	20081	2008
...			
20090101	200901	20091	2009
20090102	200901	20091	2009

EXERCISE: TEST DIMENSIONAL HIERARCHIES

UNIVERSITÀ DI PISA

Date(PkDate, Month, Quarter, Year)

How to verify on the loaded table the validity of the hierarchy **Month → Year ?**

Write a query that returns an empty result set
if **the functional dependency is valid.**

```
SELECT     Month
FROM       Date
GROUP BY   Month
HAVING     COUNT(DISTINCT Year) > 1;
```

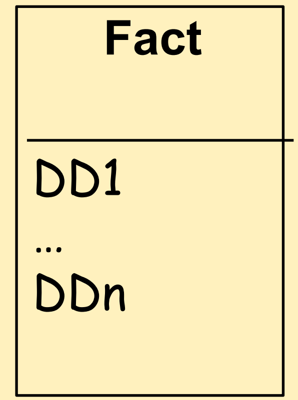
```
WITH MonthYearSubquery AS
    (SELECT DISTINCT  Month, Year
    FROM        Date)
SELECT     Month
FROM       MonthYearSubquery
GROUP BY   Month
HAVING     COUNT(*) > 1;
```

MISSING VALUES

- How to code facts where the Customer is missing?
- NULL for CustomerFK in fact table?
- Surrogate key 0 models a special customer
  - «Customer not available», «City not available», «Region not available»
- In the fact table, CustomerFK will be 0 for missing customers

DEGENERATE DIMENSIONS

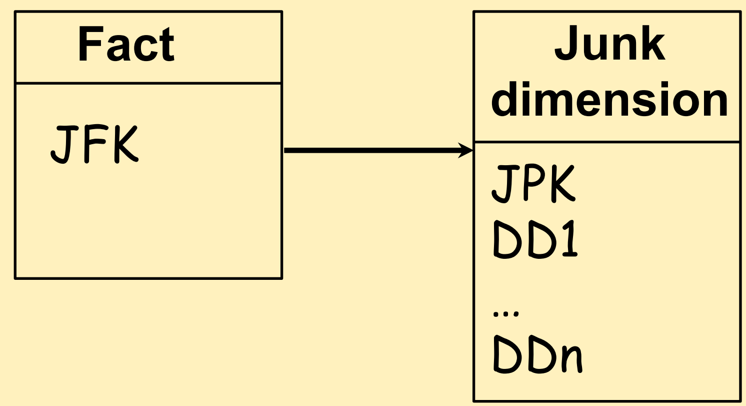
- Always stored in the fact table?
- Space to store in the fact table is
  - [space(DD1) + … + space(DDn)]*NFacts



- A junk dimension contains all possible combinations of values of DD1, …, DDn
- Space with a junk dimension is
  - space(JFK)*Nfacts +

    [space(JPK)+space(DD1) + … + space(DDn)]

    * NValues1 * … * Nvaluesn



- Which solution is more convenient?

LOGICAL DESIGN: CHANGING DIMENSIONS

Slowly changing dimensions

- **TYPE 1 (overwriting the history)** — **Overwrite the value**
  - Ex: Change the lastname Rossi instead of Rosi due to errors
- **TYPE 2 (preserving the history)** — **Add a dimension row**
  - Ex: Changing the address we do not want to lose the past ones
- **TYPE 3 (preserving one or more versions of history)** — **Add new attributes**

Not recommended

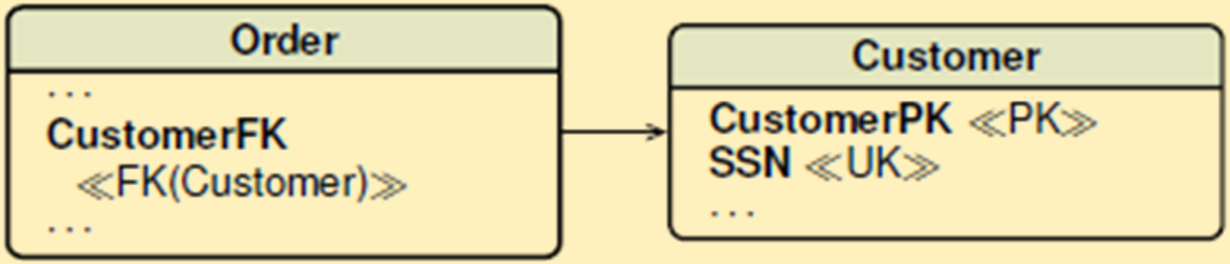
Fast changing dimensions

- **TYPE 4** — **Add a new dimension** (called **mini** or **profile**)
  - Ex: Age

These aspects are not modelled in the conceptual schema

LOGICAL DESIGN: TYPE 2 SLOWLY CHANGING DIMENSIONS

Dimensions with both a surrogate and a natural key



The customer **Jones** moved from zip code of 10019 to 45678.

CustomerPK	SSN	Name	Zip
1	31422	Murray	94025
2	**12427**	**Jones**	**10019**
3	22224	Smith	33120

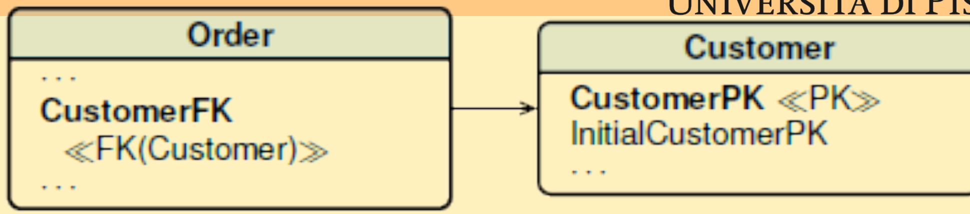
The Surrogate Key changes: more surrogate keys refer more instances of the same customers

SSN does not change

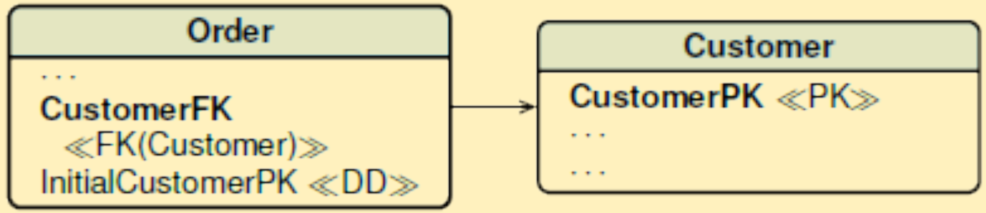
SQL: How many customer have made an Order greater than ... ?

COUNT(*) ? **Or** COUNT(DISTINCT SSN) ?

- **Dimensions with a surrogate key only**



(b) *First surrogate key in the dimension table*

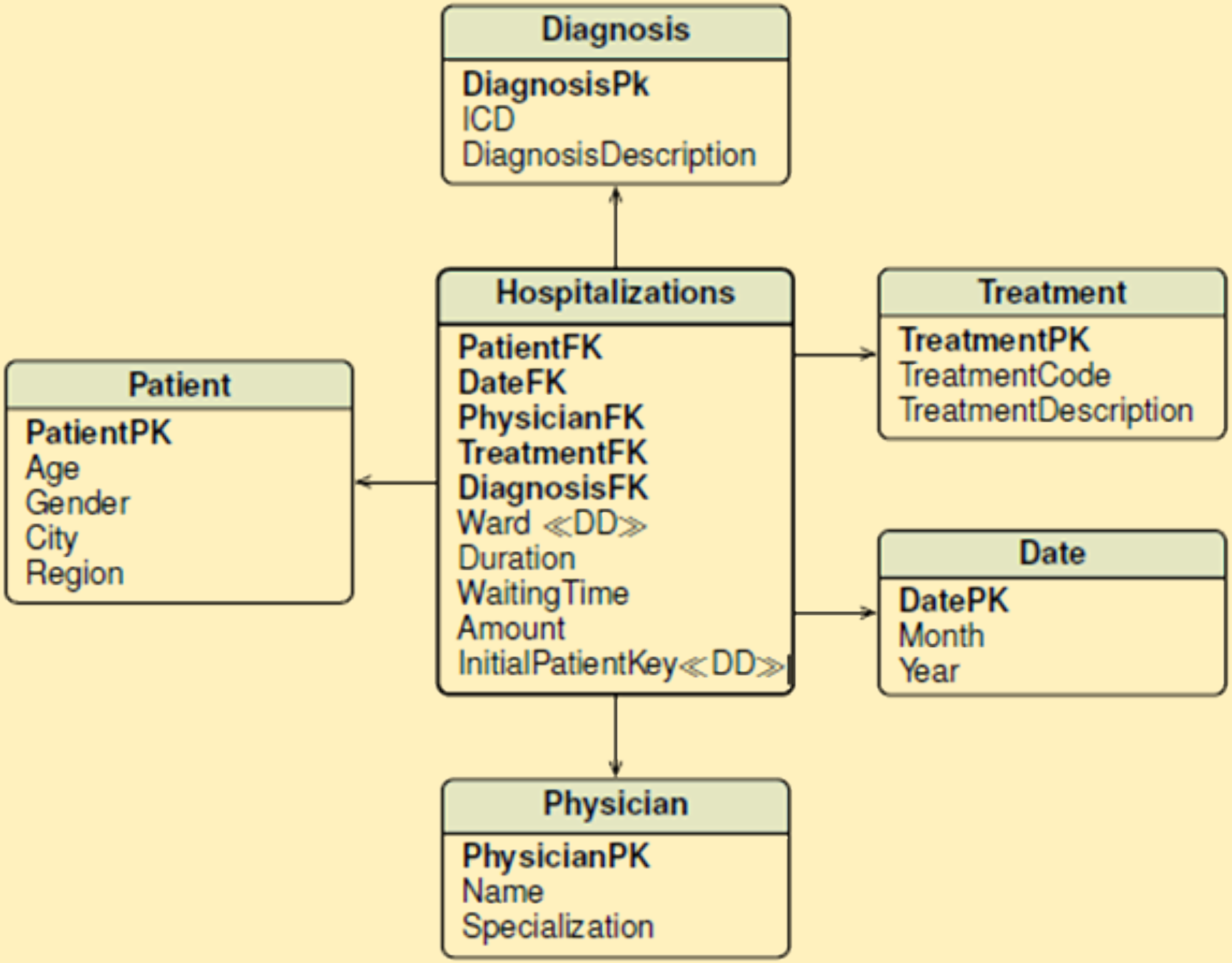


(c) *First surrogate key in the fact table*

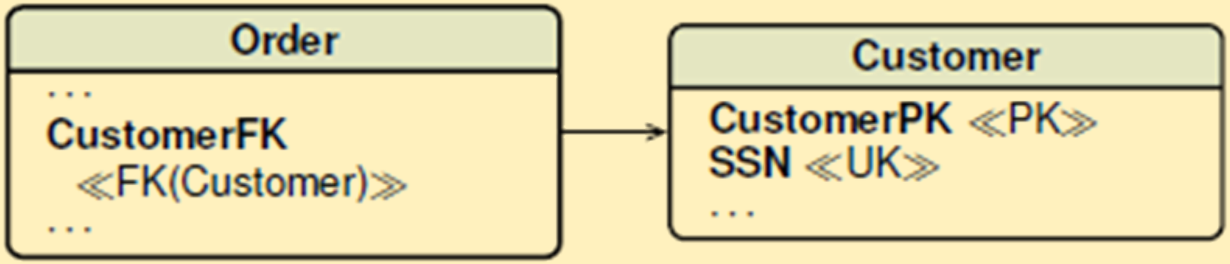
The customer **Jones** moved from zip code of 10019 to 45678.

CustomerPK	InitialCustomerPK	Name	Zip
1	1	Murray	94025
2	**2**	**Jones**	**10019**
3	3	Smith	33120

HOSPITALIZATIONS: FINAL LOGICAL SCHEMA



Add new attributes to keep track of customer data change



The customer **Jones** moved from zip code of 10019 to 45678.

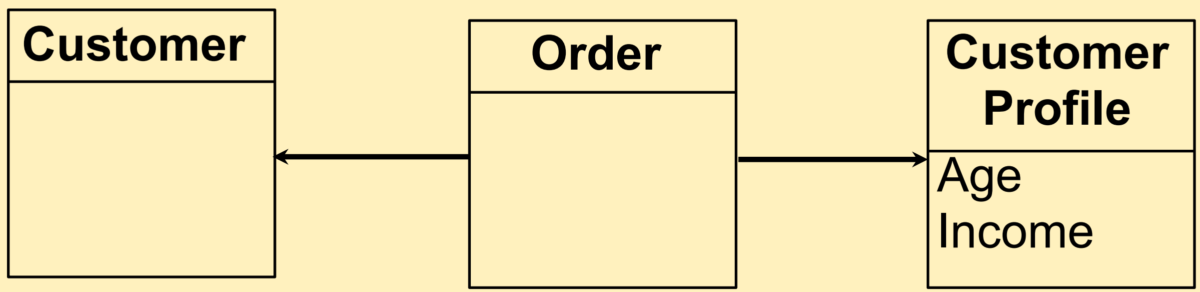
CustomerPK	SSN	Name	Zip	Old_Zip	EffDate	OldEffDate
1	31422	Murray	94025		3/1/2001	12/31/9999
2	**12427**	**Jones**	**45678**	**10019**	1/3/2008	10/10/2002
3	22224	Smith	33120		1/2/2002	12/31/9999

LOGICAL DESIGN: TYPE 4 FAST CHANGING DIMENSIONS

SMALL DIMENSIONS: Type 2 technique is still recommended

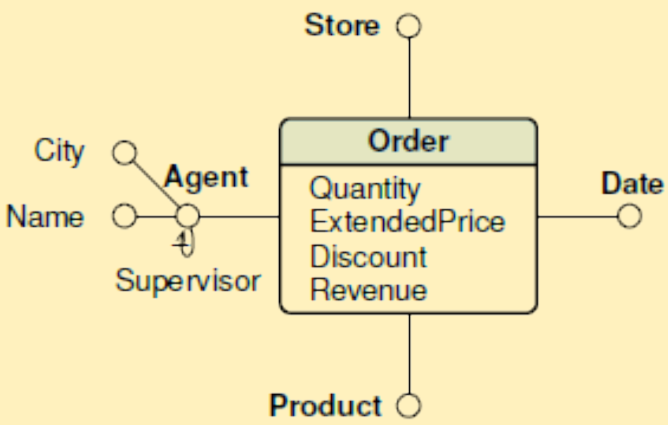
LARGE DIMENSIONS:

Create a separate dimension with frequently changing attributes



Numerical data must be converted into banded values

Insert in the new dimension all possible discrete attribute combinations at table creation time



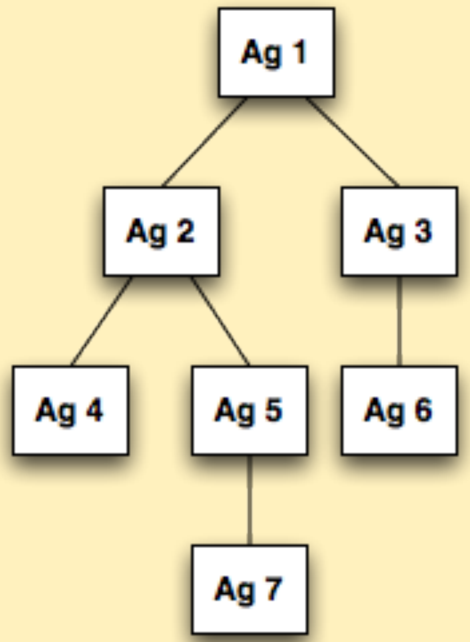
Total revenue for ***Agent* 2** and for all his subordinates

Total revenue for ***Agent* 2** and for all his supervisors

(a) *Without a bridge table*

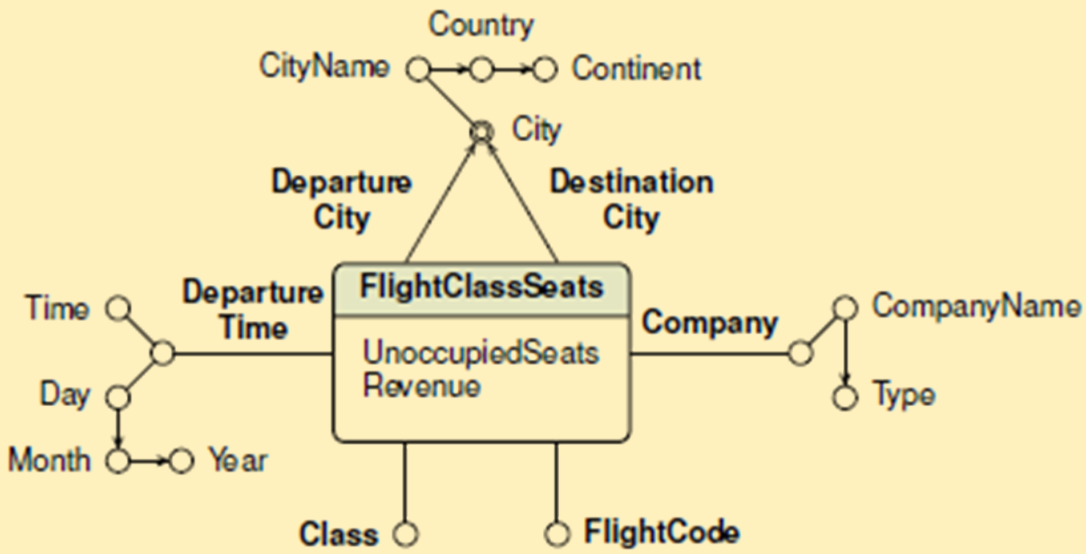
EXERCISE: WRITE THE RELATION AGENT

(a) *Without a bridge table*



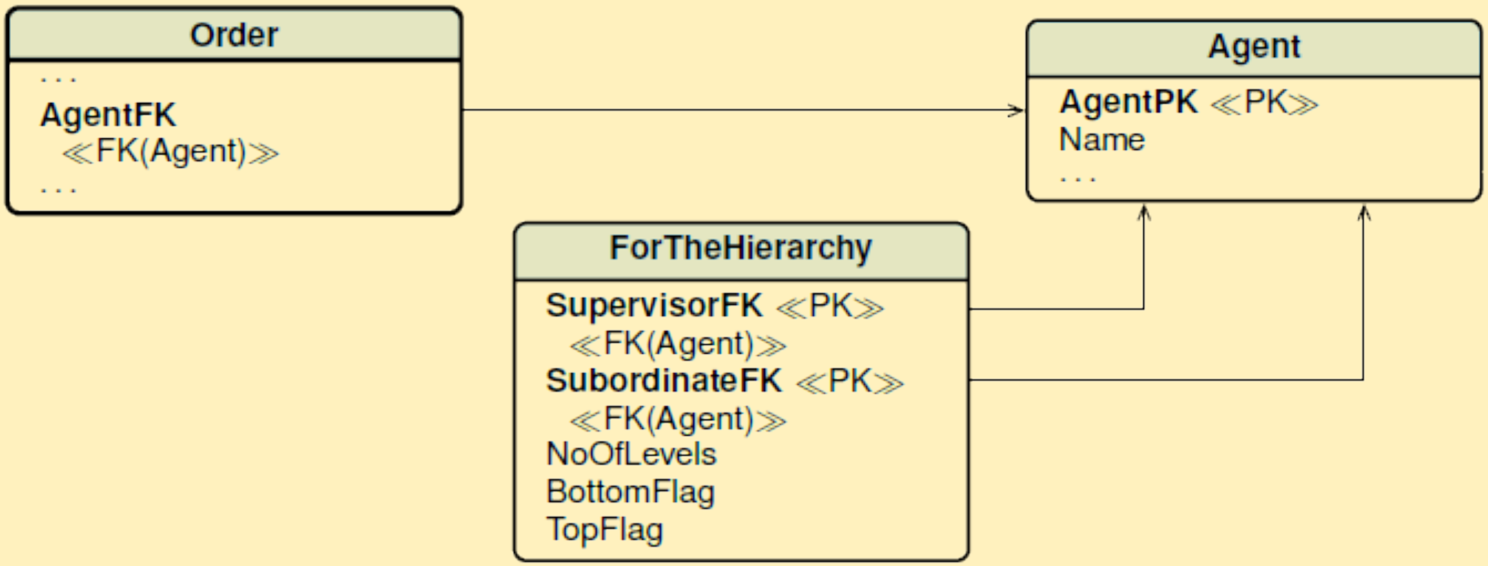
AgentPK	Name	SupervisorPK
1	Ag1	NULL
2	Ag2	1
3	Ag3	1
4	Ag4	2
5	Ag5	2
6	Ag6	3
7	Ag7	5

LOGICAL DESIGN: SHARED DIMENSIONS

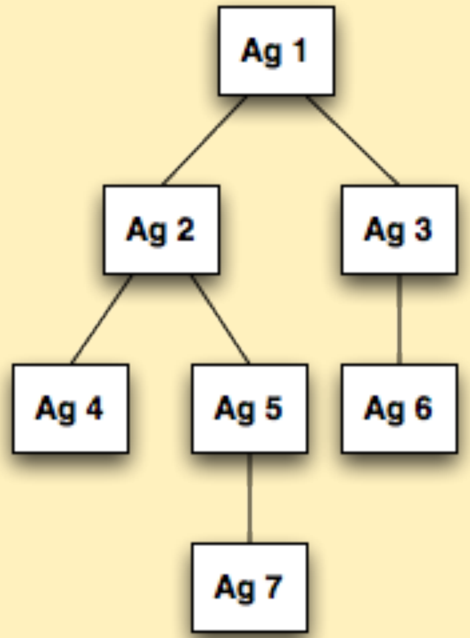


Different Hierarchies	Different tables
Shared Hierarchies	One table

LOGICAL DESIGN: RECURSIVE HIERARCHIES



(b) *With a bridge table*



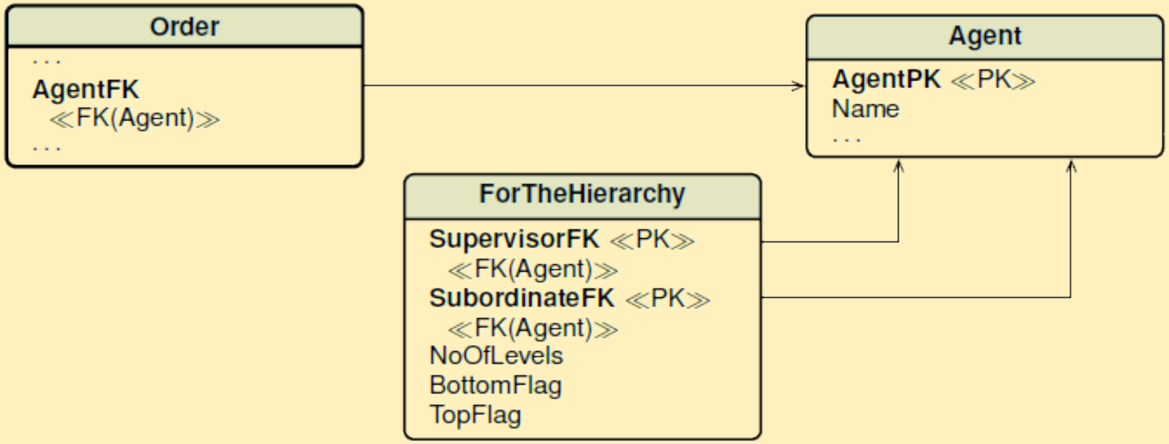
The table **ForTheHierarchy** is defined with **a record for each element of the hierarchy** plus **one for each pair (Supervisor, Subordinate)**

ForTheHierarchy

SupervisorFK	SubordinateFK	NoOfLevels	BottomFlag	TopFlag
1	1	0	F	T
1	2	1	F	F
1	3	1	F	F
1	4	2	T	F
1	5	2	F	F
1	6	2	T	F
1	7	3	T	F
2	2	0	F	F
2	4	1	T	F
2	5	1	F	F
2	7	2	T	F
3	3	0	F	F
3	6	0	T	F
4	4	0	T	F
5	5	0	F	F
5	7	1	T	F
6	6	0	T	F
7	7	0	T	F

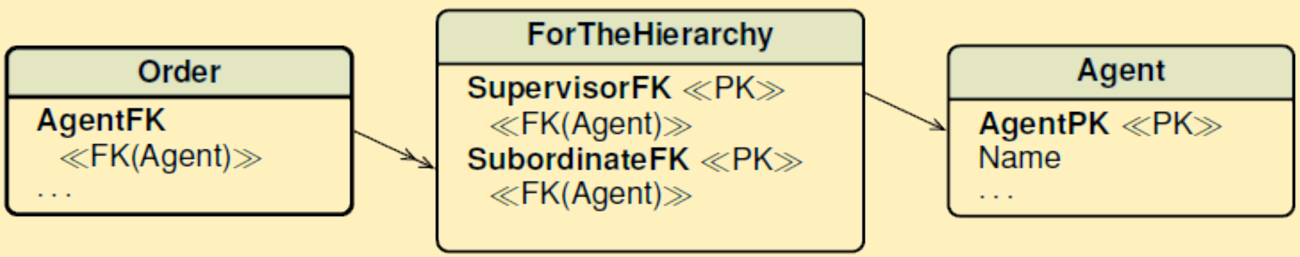
(SupervisorFK, SubordinateFK) is the Primary Key.

LOGICAL DESIGN: RECURSIVE HIERARCHIES

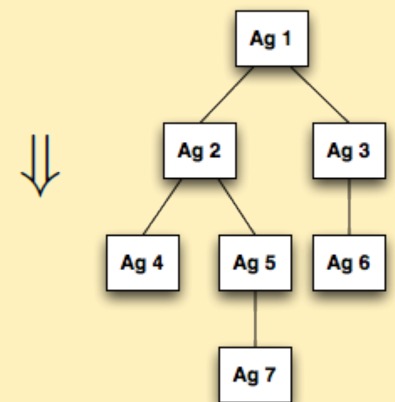


(b) *With a bridge table*

Total revenue for ***Agent 2*** and for all her subordinates



(a) *Descending the hierarchy*




```
SELECT    A.Name, SUM(Revenue)
FROM      Order O, ForTheHierarchy H, Agent A
WHERE     O.AgentFK = H.SubordinateFK AND H.SupervisorFK = A.AgentPK

GROUP BY  A.Name;
```

SUMMARY

Building a DW (conceptual and logical design, and data loading) is a **complex task** that requires **business skills**, **technology skills**, and **program management skills**.

The logical design of a conceptual schema is not trivial, especially for treating **dimensions that change over time**, **multivalued dimensions** and **multivalued dimensional attributes**.

Finally, several controls are needed for the review of a project to improve the quality of the conceptual and logical design, as described in the lecture notes.

Next, another complex task is **using** a DW to translate the business requirements into queries that can be satisfied by the DW.

OPEN LAB

- Case Studies:
  - HOSPITAL
  - AIRLINE COMPANIES
  - AIRLINE FLIGHTS
  - INVENTORY
  - HOTELS
- Design:
  - Conceptual model
  - Logical model
  - SQL queries to answer user requirements