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

<table>
<thead>
<tr>
<th>Requirements analysis</th>
<th>Dimensions</th>
<th>Measures</th>
<th>Metrics</th>
<th>Hospitalization</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## REQUIREMENTS SPECIFICATION

<table>
<thead>
<tr>
<th>Description</th>
<th>Fact granularity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preliminary dimensions</td>
<td></td>
</tr>
<tr>
<td>Preliminary measures</td>
<td></td>
</tr>
</tbody>
</table>
HOSPITALIZATIONS DATA MART CONCEPTUAL SCHEMA

DATA BASE

DATA MART

DW: Data Models, A. Albano
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

DW: Data Models, A. Albano
MORE ABOUT DATA MART CONCEPTUAL MODELLING

Degenerate dimensions

Facts descriptive attributes

Optional dimensions or attributes

Multivalued dimensions

Hierarchies types

Shared hierarchies
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

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?
CONSTELLATION SCHEMA

- **City** to **Country**
- **Market**
  - **City**
  - **State**
  - **Country**
- **Product**
  - **Name**
  - **Category**
  - **Price**
- **Sales**
  - **SalesAmt**
  - **Date**
  - **Day**
  - **Month**
  - **Year**
- **Product**
  - **ProductPK**
  - **Name**
  - **Category**
  - **Price**
- **Market**
  - **MarketPK**
  - **Name**
  - **City**
  - **State**
  - **Country**
- **Sales**
  - **MarketFK**
  - **DateFK**
  - **ProductFK**
  - **SalesAmt**
- **Returns**
  - **CustomerFK**
  - **DateFK**
  - **ProductFK**
  - **Quantity**
- **Customer**
  - **CustomerFK**
  - **Gender**
  - **Age**
  - **Income**
- **Date**
  - **DatePK**
  - **Month**
  - **Year**
THE DATE DIMENSION

Hyp: Date at daily grain

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

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Type</th>
<th>Format/Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>DatePK</td>
<td>int</td>
<td>YYYYMMDD</td>
</tr>
<tr>
<td>Month</td>
<td>int</td>
<td>YYYYMM</td>
</tr>
<tr>
<td>Quarter</td>
<td>int</td>
<td>YYYYQ</td>
</tr>
<tr>
<td>Year</td>
<td>int</td>
<td>YYYY</td>
</tr>
<tr>
<td>WeekNumber</td>
<td>int</td>
<td>1 to 52 or 53</td>
</tr>
<tr>
<td>DayInMonth</td>
<td>int</td>
<td>1 to 31</td>
</tr>
<tr>
<td>DayOfWeek</td>
<td>string</td>
<td>Monday</td>
</tr>
<tr>
<td>MonthName</td>
<td>string</td>
<td>January</td>
</tr>
<tr>
<td>HolydayName</td>
<td>string</td>
<td>Easter</td>
</tr>
</tbody>
</table>
HOSPITALIZATIONS DATA MART CONCEPTUAL SCHEMA

DESIGN THE LOGICAL SCHEMA
HOSPITALIZATIONS: INITIAL LOGICAL SCHEMA

- **Diagnosis**
  - DiagnosisPk
  - ICD
  - DiagnosisDescription

- **Patient**
  - PatientPK
  - Age
  - Gender
  - City
  - Region

- **Hospitalizations**
  - PatientFK
  - DateFK
  - PhysicianFK
  - TreatmentFK
  - DiagnosisFK
  - Ward
  - Duration
  - WaitingTime
  - Amount

- **Treatment**
  - TreatmentPK
  - TreatmentCode
  - TreatmentDescription

- **Date**
  - DatePK
  - Month
  - Year

- **Physician**
  - PhysicianPK
  - Name
  - Specialization
# AIRLINE COMPANIES: REQUIREMENTS SPECIFICATION

<table>
<thead>
<tr>
<th>Requirements analysis</th>
<th>Dimensions</th>
<th>Measure</th>
<th>Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>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)</td>
<td>FlightCode, Class, Company(Name, Type), DepartureTime (Time, Day, Month, Year)</td>
<td>UnoccupiedSeats</td>
<td>Total UnoccupiedSeats</td>
</tr>
<tr>
<td>Number of unoccupied seats in a given class and year, by flight code, by company name, by class, by departure (destination) city (country, continent).</td>
<td>FlightCode, Class, Company(Name), DepartureCity (Country, Continent), DestinationCity (Country, Continent)</td>
<td>UnoccupiedSeats</td>
<td>Total UnoccupiedSeats</td>
</tr>
<tr>
<td>Number of unoccupied seats and revenue of the Alitalia company, by year, by month, by destination country.</td>
<td>Company(Name), DepartureTime (Month, Year), DepartureCity(Country)</td>
<td>UnoccupiedSeats, Revenue</td>
<td>Total UnoccupiedSeats, Revenue</td>
</tr>
</tbody>
</table>

## Fact granularity

<table>
<thead>
<tr>
<th>Description</th>
<th>A fact is the information on the number of unoccupied seats on a flight of a class of a company</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preliminary dimensions</td>
<td>Class, FlightCode, Company, Departure time, Departure city, Destination city</td>
</tr>
<tr>
<td>Preliminary measures</td>
<td>UnoccupiedSeats, Revenue</td>
</tr>
</tbody>
</table>
AIRLINE COMPANIES: CONCEPTUAL AND LOGICAL DESIGN

[Diagram of airline companies' data model]

- Data Models, A. Albano
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?

\[ \text{Month} \rightarrow \text{Quarter} \rightarrow \text{Year} \]
LOGICAL DESIGN: STAR SCHEMA + DIMENSIONAL HIERARCHIES

PkDate $\rightarrow$ Month, Quarter, Year
Month $\rightarrow$ Quarter
Quarter $\rightarrow$ Year

Attention to the attribute values!

<table>
<thead>
<tr>
<th>Date</th>
<th>PkDate</th>
<th>Month</th>
<th>Quarter</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>20080101</td>
<td>200801</td>
<td>20081</td>
<td>2008</td>
<td></td>
</tr>
<tr>
<td>20080102</td>
<td>200801</td>
<td>20081</td>
<td>2008</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20090101</td>
<td>200901</td>
<td>20091</td>
<td>2009</td>
<td></td>
</tr>
<tr>
<td>20090102</td>
<td>200901</td>
<td>20091</td>
<td>2009</td>
<td></td>
</tr>
</tbody>
</table>
EXERCISE: TEST DIMENSIONAL HIERARCHIES

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.

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
  - \([\text{space}(DD1) + \ldots + \text{space}(DDn)]*N\text{Facts}\)
- A junk dimension contains all possible combinations of values of DD1, ..., DDn
- Space with a junk dimension is
  - \(\text{space}(JFK)*N\text{facts} +\)
    - \([\text{space}(JPK)+\text{space}(DD1) + \ldots + \text{space}(DDn)]\)
    - \(*N\text{Values}_1 * \ldots * N\text{Values}_n\)
- Which solution is more convenient?
LOGICAL DESIGN: CHANGING DIMENSIONS

Slowly changing dimensions

- TYPE 1 (overwriting the history)
  - Ex: Change the lastname Rossi instead of Rosi due to errors
- TYPE 2 (preserving the history)
  - Ex: Changing the address we do not want to lose the past ones
- TYPE 3 (preserving one or more versions of history)
  - Not recommended

Fast changing dimensions

- TYPE 4
  - Ex: Age

These aspects are not modelled in the conceptual schema

Overwrite the value
Add a dimension row
Add new attributes
Add a new dimension (called mini or profile)
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.

<table>
<thead>
<tr>
<th>CustomerPK</th>
<th>SSN</th>
<th>Name</th>
<th>Zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>31422</td>
<td>Murray</td>
<td>94025</td>
</tr>
<tr>
<td>2</td>
<td>12427</td>
<td>Jones</td>
<td>10019</td>
</tr>
<tr>
<td>3</td>
<td>22224</td>
<td>Smith</td>
<td>33120</td>
</tr>
</tbody>
</table>

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) ?
```
LOGICAL DESIGN: TYPE 2 SLOWLY CHANGING DIMENSIONS

• Dimensions with a surrogate key only

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

<table>
<thead>
<tr>
<th>CustomerPK</th>
<th>InitialCustomerPK</th>
<th>Name</th>
<th>Zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Murray</td>
<td>94025</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>Jones</td>
<td>10019</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>Smith</td>
<td>33120</td>
</tr>
</tbody>
</table>
Add new attributes to keep track of customer data change

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

<table>
<thead>
<tr>
<th>CustomerPK</th>
<th>SSN</th>
<th>Name</th>
<th>Zip</th>
<th>Old_Zip</th>
<th>EffDate</th>
<th>OldEffDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>31422</td>
<td>Murray</td>
<td>94025</td>
<td></td>
<td>3/1/2001</td>
<td>12/31/9999</td>
</tr>
<tr>
<td>2</td>
<td>12427</td>
<td>Jones</td>
<td>45678</td>
<td>10019</td>
<td>1/3/2008</td>
<td>10/10/2002</td>
</tr>
<tr>
<td>3</td>
<td>22224</td>
<td>Smith</td>
<td>33120</td>
<td></td>
<td>1/2/2002</td>
<td>12/31/9999</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>AgentPK</th>
<th>Name</th>
<th>SupervisorPK</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ag1</td>
<td>NULL</td>
</tr>
<tr>
<td>2</td>
<td>Ag2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>Ag3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>Ag4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>Ag5</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>Ag6</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>Ag7</td>
<td>5</td>
</tr>
</tbody>
</table>
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).

(SupervisorFK, SubordinateFK) is the Primary Key.
Total revenue for Agent 2 and for all her subordinates

(b) With a bridge table

(a) Descending the hierarchy

```sql
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