Dimensional Modeling

Krzysztof Dembczyński

Intelligent Decision Support Systems Laboratory (IDSS)
Poznań University of Technology, Poland

Bachelor studies, seventh semester
Academic year 2018/19 (winter semester)

Review of the Previous Lecture

- Processing of massive datasets.
- Evolution of database systems:
  - Operational (OLTP) vs. analytical (OLAP) systems.
  - Analytical database systems.
  - Design of data warehouses.
  - Relational model vs. multidimensional model.
  - NoSQL.
  - Processing of massive datasets.

Outline

Outline

Motivation

- University authorities decided to analyze teaching performance by using the data collected in databases owned by the university containing information about students, instructors, lectures, faculties, etc.

Motivation

- University authorities decided to analyze teaching performance by using the data collected in databases owned by the university containing information about students, instructors, lectures, faculties, etc.
- They would like to get answers for the following queries:

Motivation

- University authorities decided to analyze teaching performance by using the data collected in databases owned by the university containing information about students, instructors, lectures, faculties, etc.
- They would like to get answers for the following queries:

Motivation

- University authorities decided to analyze teaching performance by using the data collected in databases owned by the university containing information about students, instructors, lectures, faculties, etc.
- They would like to get answers for the following queries:
  - What is the average score of students over academic years?

Motivation

- University authorities decided to analyze teaching performance by using the data collected in databases owned by the university containing information about students, instructors, lectures, faculties, etc.
- They would like to get answers for the following queries:
  - What is the average score of students over academic years?
  - What is the number of students over academic years?

Motivation

- University authorities decided to analyze teaching performance by using the data collected in databases owned by the university containing information about students, instructors, lectures, faculties, etc.
- They would like to get answers for the following queries:
  - What is the average score of students over academic years?
  - What is the number of students over academic years?
  - What is the average score by faculties, instructors, etc.?

Motivation

- University authorities decided to analyze teaching performance by using the data collected in databases owned by the university containing information about students, instructors, lectures, faculties, etc.
- They would like to get answers for the following queries:
  - What is the average score of students over academic years?
  - What is the number of students over academic years?
  - What is the average score by faculties, instructors, etc.?
  - What is the distribution of students over faculties, semesters, etc.?

Motivation

- University authorities decided to analyze teaching performance by using the data collected in databases owned by the university containing information about students, instructors, lectures, faculties, etc.
- They would like to get answers for the following queries:
  - What is the average score of students over academic years?
  - What is the number of students over academic years?
  - What is the average score by faculties, instructors, etc.?
  - What is the distribution of students over faculties, semesters, etc.?
  - ...

Example

- An exemplary query could be the following:

```
SELECT Instructor, Academic_year, AVG(Grade)
FROM Data_Warehouse
GROUP BY Instructor, Academic_year
```

- And the result:

Academic_year	Name	AVG(Grade)
2010/11	Stefanowski	4.2
2011/12	Stefanowski	4.5
2010/12	Słowiński	4.3
2011/12	Słowiński	4.1
2011/12	Dembczyński	:)

Motivation

- **Problem to solve**: How to design a database for analytical queries?

Outline

Conceptual schemes of data warehouses

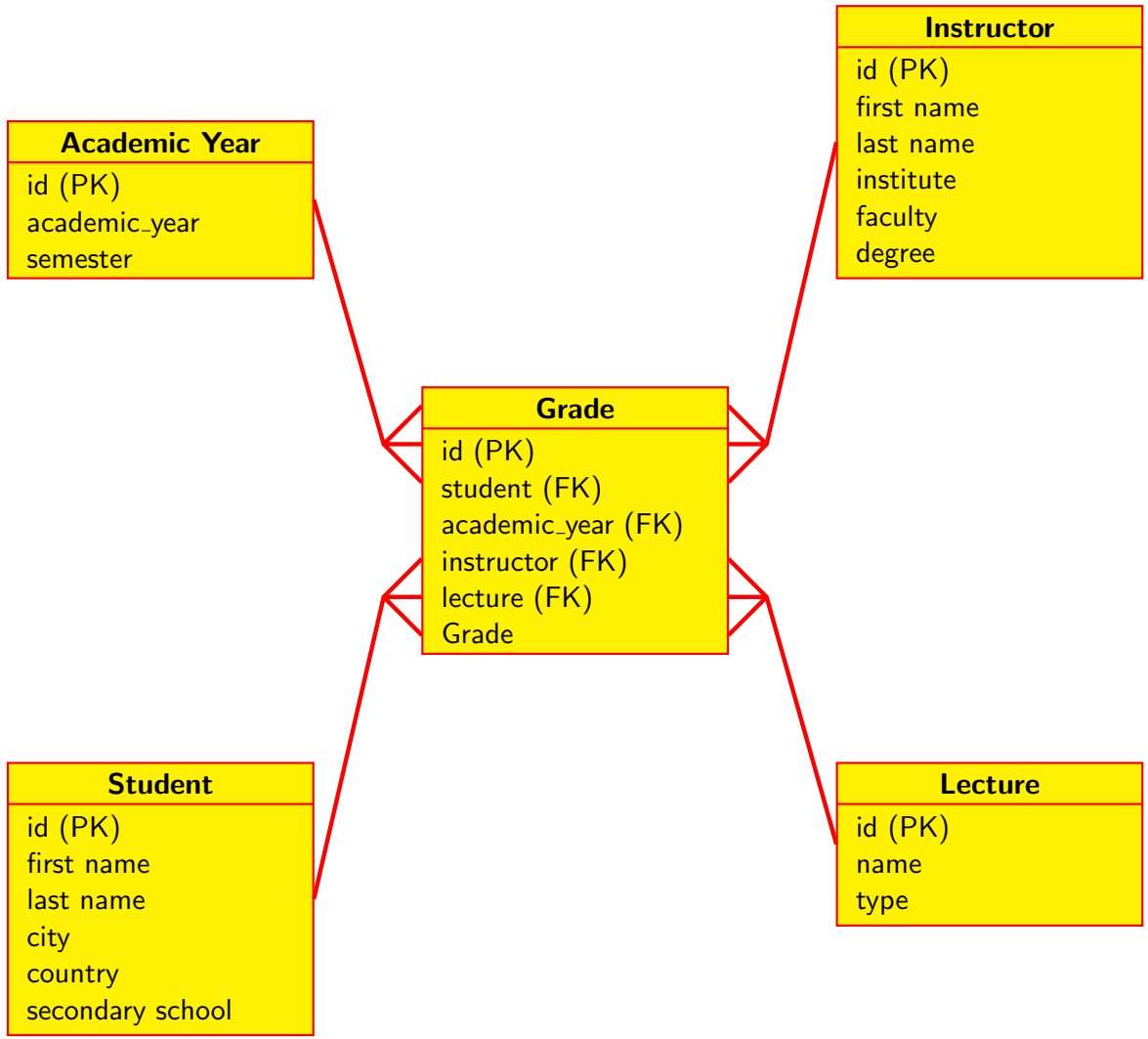
- Three main goals for logical design:
  - Simplicity:
    - Users should understand the design,
    - Data model should match users' conceptual model,
    - Queries should be easy and intuitive to write.
  - Expressiveness:
    - Include enough information to answer all important queries,
    - Include all relevant data (without irrelevant data).
  - Performance:
    - An efficient physical design should be possible to apply.

Three basic conceptual schemes

- Star schema,
- Snowflake schema,
- Fact constellations.

Star schema

- A single table in the middle connected to a number of dimension tables.



Star schema

- **Measures**, e.g. grades, price, quantity.

Star schema

- **Measures**, e.g. grades, price, quantity.
  - Measures should be aggregative.

Star schema

- **Measures**, e.g. grades, price, quantity.
  - Measures should be aggregative.
  - Measures depend on a set of dimensions, e.g. student grade depends on student, course, instructor, faculty, academic year, etc..

Star schema

- **Measures**, e.g. grades, price, quantity.
  - Measures should be aggregative.
  - Measures depend on a set of dimensions, e.g. student grade depends on student, course, instructor, faculty, academic year, etc..
- **Fact table**

Star schema

- **Measures**, e.g. grades, price, quantity.
  - Measures should be aggregative.
  - Measures depend on a set of dimensions, e.g. student grade depends on student, course, instructor, faculty, academic year, etc..
- **Fact table**
  - Relates the dimensions to the measures.

Star schema

- **Measures**, e.g. grades, price, quantity.
  - Measures should be aggregative.
  - Measures depend on a set of dimensions, e.g. student grade depends on student, course, instructor, faculty, academic year, etc..
- **Fact table**
  - Relates the dimensions to the measures.
- **Dimension tables**

Star schema

- **Measures**, e.g. grades, price, quantity.
  - Measures should be aggregative.
  - Measures depend on a set of dimensions, e.g. student grade depends on student, course, instructor, faculty, academic year, etc..
- **Fact table**
  - Relates the dimensions to the measures.
- **Dimension tables**
  - Represent information about dimensions (student, academic year, etc.).

Star schema

- **Measures**, e.g. grades, price, quantity.
  - Measures should be aggregative.
  - Measures depend on a set of dimensions, e.g. student grade depends on student, course, instructor, faculty, academic year, etc..
- **Fact table**
  - Relates the dimensions to the measures.
- **Dimension tables**
  - Represent information about dimensions (student, academic year, etc.).
  - Each dimension has a set of descriptive attributes.

Fact table

- Each fact table contains measurements about a process of interest.

Fact table

- Each fact table contains measurements about a process of interest.
- Each fact row contains foreign keys to dimension tables and numerical measure columns.

Fact table

- Each fact table contains measurements about a process of interest.
- Each fact row contains foreign keys to dimension tables and numerical measure columns.
- Any new fact is added to the fact table.

Fact table

- Each fact table contains measurements about a process of interest.
- Each fact row contains foreign keys to dimension tables and numerical measure columns.
- Any new fact is added to the fact table.
- The aggregated fact columns are the matter of the analysis.

Dimension tables

- Each dimension table corresponds to a real-world object or concept, e.g. customer, product, region, employee, store, etc..

Dimension tables

- Each dimension table corresponds to a real-world object or concept, e.g. customer, product, region, employee, store, etc..
- Dimension tables contain many descriptive columns.

Dimension tables

- Each dimension table corresponds to a real-world object or concept, e.g. customer, product, region, employee, store, etc..
- Dimension tables contain many descriptive columns.
- Generally do not have too many rows (in comparison to the fact table).

Dimension tables

- Each dimension table corresponds to a real-world object or concept, e.g. customer, product, region, employee, store, etc..
- Dimension tables contain many descriptive columns.
- Generally do not have too many rows (in comparison to the fact table).
- Content is relatively static.

Dimension tables

- Each dimension table corresponds to a real-world object or concept, e.g. customer, product, region, employee, store, etc..
- Dimension tables contain many descriptive columns.
- Generally do not have too many rows (in comparison to the fact table).
- Content is relatively static.
- The attributes of dimension tables are used for filtering and grouping.

Dimension tables

- Each dimension table corresponds to a real-world object or concept, e.g. customer, product, region, employee, store, etc..
- Dimension tables contain many descriptive columns.
- Generally do not have too many rows (in comparison to the fact table).
- Content is relatively static.
- The attributes of dimension tables are used for filtering and grouping.
- Dimension tables describe facts stored in the fact table.

Fact table vs. Dimension tables

- Fact table:

Facts contain numbers, dimensions contain labels

Fact table vs. Dimension tables

- Fact table:
  - narrow,

Facts contain numbers, dimensions contain labels

Fact table vs. Dimension tables

- Fact table:
  - narrow,
  - big (many rows),

Facts contain numbers, dimensions contain labels

Fact table vs. Dimension tables

- Fact table:
  - narrow,
  - big (many rows),
  - numeric (rows are described by numerical measures),

Facts contain numbers, dimensions contain labels

Fact table vs. Dimension tables

- Fact table:
  - narrow,
  - big (many rows),
  - numeric (rows are described by numerical measures),
  - dynamic (growing over time).

Facts contain numbers, dimensions contain labels

Fact table vs. Dimension tables

- Fact table:
  - narrow,
  - big (many rows),
  - numeric (rows are described by numerical measures),
  - dynamic (growing over time).
- Dimension table

Facts contain numbers, dimensions contain labels

Fact table vs. Dimension tables

- Fact table:
  - narrow,
  - big (many rows),
  - numeric (rows are described by numerical measures),
  - dynamic (growing over time).
- Dimension table
  - wide,

Facts contain numbers, dimensions contain labels

Fact table vs. Dimension tables

- Fact table:
  - narrow,
  - big (many rows),
  - numeric (rows are described by numerical measures),
  - dynamic (growing over time).
- Dimension table
  - wide,
  - small (few rows),

Facts contain numbers, dimensions contain labels

Fact table vs. Dimension tables

- Fact table:
  - narrow,
  - big (many rows),
  - numeric (rows are described by numerical measures),
  - dynamic (growing over time).
- Dimension table
  - wide,
  - small (few rows),
  - descriptive (rows are described by descriptive attributes),

Facts contain numbers, dimensions contain labels

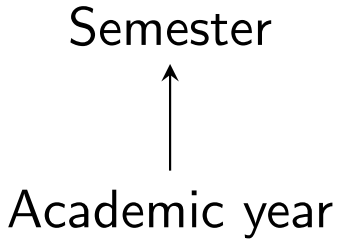
Fact table vs. Dimension tables

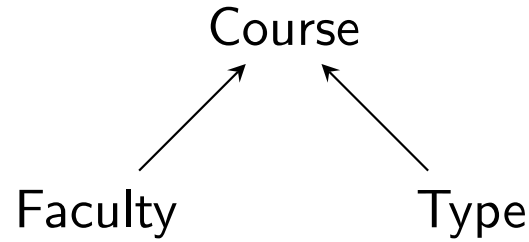
- Fact table:
  - narrow,
  - big (many rows),
  - numeric (rows are described by numerical measures),
  - dynamic (growing over time).
- Dimension table
  - wide,
  - small (few rows),
  - descriptive (rows are described by descriptive attributes),
  - static.

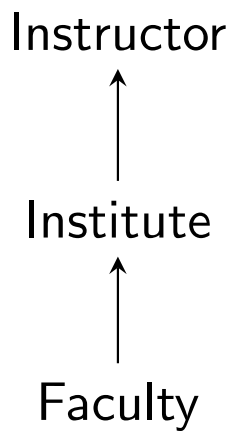
Facts contain numbers, dimensions contain labels

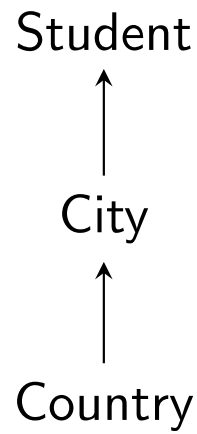
Dimension hierarchies

- For each dimension, one can define several hierarchies of attributes.



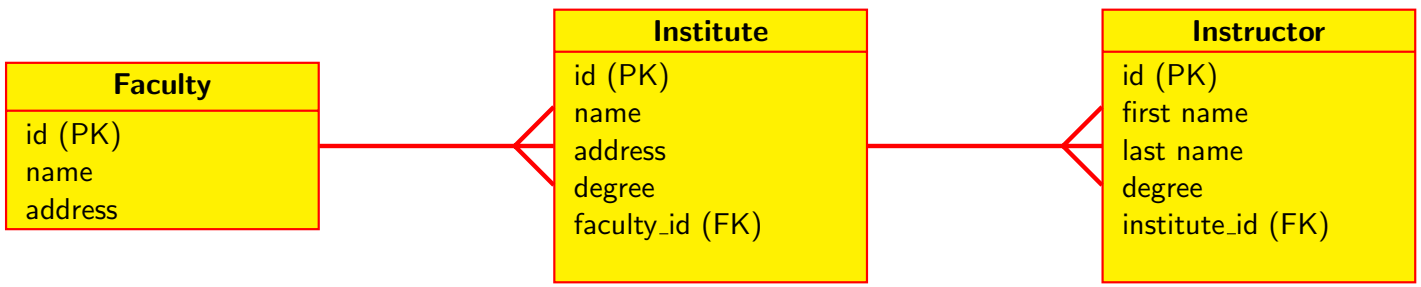






Snowflake schema

- Snowflake schema is a refinement of star schema where the dimensional hierarchy is represented explicitly by normalizing the dimension tables.



Normalization

- **Normal forms** provide criteria for determining a table's degree of vulnerability to inconsistencies, redundancy and update anomalies.
  - First normal form (1NF),
  - Second normal form (2NF),
  - Third normal from (3NF),
  - Boyce-Codd normal form (BCNF),
  - Fourth normal form (4NF),
  - ...

Denormalization

- Denormalization is the process of attempting to optimize the performance of a database by adding redundant data or by grouping data.
- Denormalization helps cover up the inefficiencies inherent in relational database software.
- **Normalize until it hurts, denormalize until it works** :)

Fact constellation schema

- Fact constellation schema consists of multiple fact tables which share dimension tables.

Fact constellation schema

- Fact constellation schema consists of multiple fact tables which share dimension tables.
- Such schema appears in a design of a data warehouse for a huge and complex problem.

Fact constellation schema

- Fact constellation schema consists of multiple fact tables which share dimension tables.
- Such schema appears in a design of a data warehouse for a huge and complex problem.
- One should start with a simple model, and then try to define a more complex one (project success).

Outline

Dimensional modeling

- Four Steps in Dimensional Modeling:

Dimensional modeling

- Four Steps in Dimensional Modeling:
  - Select the business process to model (e.g. sales).

Dimensional modeling

- Four Steps in Dimensional Modeling:
  - Select the business process to model (e.g. sales).
  - Determine the grain of the business process (e.g. single transaction in a market identified by bar-code scanners at cash register).

Dimensional modeling

- Four Steps in Dimensional Modeling:
  - Select the business process to model (e.g. sales).
  - Determine the grain of the business process (e.g. single transaction in a market identified by bar-code scanners at cash register).
  - Choose the dimensions describing the business process (e.g. localization, product, data, promotion, etc.).

Dimensional modeling

- Four Steps in Dimensional Modeling:
  - Select the business process to model (e.g. sales).
  - Determine the grain of the business process (e.g. single transaction in a market identified by bar-code scanners at cash register).
  - Choose the dimensions describing the business process (e.g. localization, product, data, promotion, etc.).
  - Identify the numeric measures for the facts (e.g. price, quantity).

Business process

- Business process has to be a natural activity performed in the given organization that is supported by an operational database system.

Business process

- Business process has to be a natural activity performed in the given organization that is supported by an operational database system.
- **Examples**: sales, orders, teaching.

Business process

- Business process has to be a natural activity performed in the given organization that is supported by an operational database system.
- **Examples**: sales, orders, teaching.
- Business process should not be referred to an organizational business department or function.

Business process

- Business process has to be a natural activity performed in the given organization that is supported by an operational database system.
- **Examples**: sales, orders, teaching.
- Business process should not be referred to an organizational business department or function.
- Selection of the business process should be dependent on its complexity, and also on time and budget of the project.

Grain of the business process

- Define what an individual fact table row represents.

Grain of the business process

- Define what an individual fact table row represents.
- Determine the maximum level of detail of the warehouse.

Grain of the business process

- Define what an individual fact table row represents.
- Determine the maximum level of detail of the warehouse.
- **Examples**: line item from a cash register receipt, daily snapshot of inventory level for a product in a warehouse, monthly snapshot for each bank account.

Grain of the business process

- Define what an individual fact table row represents.
- Determine the maximum level of detail of the warehouse.
- **Examples**: line item from a cash register receipt, daily snapshot of inventory level for a product in a warehouse, monthly snapshot for each bank account.
- Finer-grained fact tables are more expressive, but have more rows and grow faster.

Dimensions describing the business process

- Decorate fact table with a set of dimensions that determine a candidate key based on the grain and all other dimensions functionally determined by the candidate key.

Dimensions describing the business process

- Decorate fact table with a set of dimensions that determine a candidate key based on the grain and all other dimensions functionally determined by the candidate key.
- In other words: detailed description of the grain defined in the previous step.

Dimensions describing the business process

- Decorate fact table with a set of dimensions that determine a candidate key based on the grain and all other dimensions functionally determined by the candidate key.
- In other words: detailed description of the grain defined in the previous step.
- Grain of the fact table defines the grain of the dimensions.

Dimensions describing the business process

- Decorate fact table with a set of dimensions that determine a candidate key based on the grain and all other dimensions functionally determined by the candidate key.
- In other words: detailed description of the grain defined in the previous step.
- Grain of the fact table defines the grain of the dimensions.
- **Examples**: localization, product, date, promotion, etc.

Numeric measures

- Numeric measures allow to measure the performance of the process.

Numeric measures

- Numeric measures allow to measure the performance of the process.
- All the measures correspond to the same grain defined previously.

Numeric measures

- Numeric measures allow to measure the performance of the process.
- All the measures correspond to the same grain defined previously.
- Measures should be numeric and additive:

Numeric measures

- Numeric measures allow to measure the performance of the process.
- All the measures correspond to the same grain defined previously.
- Measures should be numeric and additive:
  - fully additive: can be summed across any of the dimensions associated with the fact table

Numeric measures

- Numeric measures allow to measure the performance of the process.
- All the measures correspond to the same grain defined previously.
- Measures should be numeric and additive:
  - fully additive: can be summed across any of the dimensions associated with the fact table
  - semi-additive: can be summed across, but not all, for example balance amounts are additive across all dimensions except time.

Numeric measures

- Numeric measures allow to measure the performance of the process.
- All the measures correspond to the same grain defined previously.
- Measures should be numeric and additive:
  - fully additive: can be summed across any of the dimensions associated with the fact table
  - semi-additive: can be summed across, but not all, for example balance amounts are additive across all dimensions except time.
  - non-additive: are completely non-additive, such as ratios (store the fully additive components of the non-additive measure and sum these components into the final answer before calculating the final non-additive fact)

Dimensional modeling techniques

- Derived measures vs. calculated measures,
- Describing dimensions,
- Date and time dimension,
- Surrogate keys,
- Degenerate dimensions,
- Role-playing dimensions,
- Junk dimension,
- Slowly changing dimensions,
- Mini dimension,
- Factless fact tables,
- Data warehouse bus architecture.

Derived measures vs. Calculated measures

- If possible materialize (calculate and store) all measures needed for analytical queries.

Derived measures vs. Calculated measures

- If possible materialize (calculate and store) all measures needed for analytical queries.
- Data warehouse is huge from the definition, so storing additional values does not change a lot.

Derived measures vs. Calculated measures

- If possible materialize (calculate and store) all measures needed for analytical queries.
- Data warehouse is huge from the definition, so storing additional values does not change a lot.
- Computing additional measures (like profit from incomes and outcomes) decreases performance of data warehouse.

Describing dimension

- Always use informative names for tables, attributes, and values,

Describing dimension

- Always use informative names for tables, attributes, and values,
- Use as many as possible, but useful, dimension attributes

Date dimension

- Date dimension is a specific and inseparable dimension of the data warehouse design.

Date dimension

- Date dimension is a specific and inseparable dimension of the data warehouse design.
- Data warehouse should be treated as a temporal database.

Date dimension

- Date dimension is a specific and inseparable dimension of the data warehouse design.
- Data warehouse should be treated as a temporal database.
- Date dimension allows analysis of time trends.

Date dimension

- Date dimension is a specific and inseparable dimension of the data warehouse design.
- Data warehouse should be treated as a temporal database.
- Date dimension allows analysis of time trends.
- Should we treat date as a separate dimension or use timestamps to compute all needed information?

Date dimension

- Date dimension is a specific and inseparable dimension of the data warehouse design.
- Data warehouse should be treated as a temporal database.
- Date dimension allows analysis of time trends.
- Should we treat date as a separate dimension or use timestamps to compute all needed information?
  - A separate dimension is more reasonable ...

Date dimension

- Typical attributes in date dimension:
  - date and time (candidate key),
  - day (number) of year,
  - day (number) of month,
  - day (number) of week,
  - day number in epoch,
  - weekday,
  - weekend,
  - 24-hour work day
  - holiday
  - week of year,
  - month,
  - name of month,
  - quarter,
  - year,
  - fiscal year,
  - selling season,
  - back-to-school
  - Christmas

Date dimension

- Advantages of using date dimension:

Date dimension

- Advantages of using date dimension:
  - Capturing interesting date properties (holiday/non-holiday),

Date dimension

- Advantages of using date dimension:
  - Capturing interesting date properties (holiday/non-holiday),
  - Avoiding relying on special date-handling SQL functions,

Date dimension

- Advantages of using date dimension:
  - Capturing interesting date properties (holiday/non-holiday),
  - Avoiding relying on special date-handling SQL functions,
  - Making use of indexes.

Date dimension

- Advantages of using date dimension:
  - Capturing interesting date properties (holiday/non-holiday),
  - Avoiding relying on special date-handling SQL functions,
  - Making use of indexes.
- For finer-grained time measurements, use separate date and time-of-day

Surrogate keys

- A surrogate key is a meaningless integer key that is assigned by the data warehouse.

Surrogate keys

- A surrogate key is a meaningless integer key that is assigned by the data warehouse.
- Surrogate keys should be used instead of natural keys (like PESEL number in Poland):

Surrogate keys

- A surrogate key is a meaningless integer key that is assigned by the data warehouse.
- Surrogate keys should be used instead of natural keys (like PESEL number in Poland):
  - Surrogate keys can be shorter.

Surrogate keys

- A surrogate key is a meaningless integer key that is assigned by the data warehouse.
- Surrogate keys should be used instead of natural keys (like PESEL number in Poland):
  - Surrogate keys can be shorter.
  - Better handling of exceptional cases (e.g. for missing values – try to use a specific row in the dimension table that represents missing value).

Surrogate keys

- A surrogate key is a meaningless integer key that is assigned by the data warehouse.
- Surrogate keys should be used instead of natural keys (like PESEL number in Poland):
  - Surrogate keys can be shorter.
  - Better handling of exceptional cases (e.g. for missing values – try to use a specific row in the dimension table that represents missing value).
  - Surrogate keys avoid to assume implicit semantic.

Surrogate keys

- A surrogate key is a meaningless integer key that is assigned by the data warehouse.
- Surrogate keys should be used instead of natural keys (like PESEL number in Poland):
  - Surrogate keys can be shorter.
  - Better handling of exceptional cases (e.g. for missing values – try to use a specific row in the dimension table that represents missing value).
  - Surrogate keys avoid to assume implicit semantic.
  - Resistant to change of natural key meaning.

Surrogate keys

- A surrogate key is a meaningless integer key that is assigned by the data warehouse.
- Surrogate keys should be used instead of natural keys (like PESEL number in Poland):
  - Surrogate keys can be shorter.
  - Better handling of exceptional cases (e.g. for missing values – try to use a specific row in the dimension table that represents missing value).
  - Surrogate keys avoid to assume implicit semantic.
  - Resistant to change of natural key meaning.
  - Resistant to reuse of old value of natural key.

Degenerate dimensions

- Dimension can be merely an identifier, without any interesting attributes:

Degenerate dimensions

- Dimension can be merely an identifier, without any interesting attributes:
  - Consider a data warehouses for a retail market,

Degenerate dimensions

- Dimension can be merely an identifier, without any interesting attributes:
  - Consider a data warehouses for a retail market,
  - Typical transaction can be described by (id transaction, product, ...),

Degenerate dimensions

- Dimension can be merely an identifier, without any interesting attributes:
  - Consider a data warehouses for a retail market,
  - Typical transaction can be described by (id transaction, product, ...),
  - id transaction is just a unique identifier.

Degenerate dimensions

- Dimension can be merely an identifier, without any interesting attributes:
  - Consider a data warehouses for a retail market,
  - Typical transaction can be described by (id transaction, product, ...),
  - id transaction is just a unique identifier.
- There are two options for dealing with such dimensions:

Degenerate dimensions

- Dimension can be merely an identifier, without any interesting attributes:
  - Consider a data warehouses for a retail market,
  - Typical transaction can be described by (id transaction, product, ...),
  - id transaction is just a unique identifier.
- There are two options for dealing with such dimensions:
  - Discard the dimension,

Degenerate dimensions

- Dimension can be merely an identifier, without any interesting attributes:
  - Consider a data warehouses for a retail market,
  - Typical transaction can be described by (id transaction, product, ...),
  - id transaction is just a unique identifier.
- There are two options for dealing with such dimensions:
  - Discard the dimension,
  - Use a **degenerate dimension** ...

Degenerate dimensions

- Store the dimension identifier directly in the fact table.

Degenerate dimensions

- Store the dimension identifier directly in the fact table.
- Do not create a separate dimension table.

Degenerate dimensions

- Store the dimension identifier directly in the fact table.
- Do not create a separate dimension table.
- Such a dimension serves to group together products that were bought in the same market basket (market basket analysis).

Role-playing dimensions

- A single physical dimension can be referenced multiple times in a fact table, with each reference linking to a logically distinct role for the dimension.

Role-playing dimensions

- A single physical dimension can be referenced multiple times in a fact table, with each reference linking to a logically distinct role for the dimension.
- **Example**: a fact is associated with several dates (e.g. order, payment, or delivery date), each of which is represented by a foreign key.

Junk dimension

- Some columns may not fit to any of the dimensions, but creating a separate dimension for each of them would increase significantly the size of fact table (a foreign key would be required for each such column).

Junk dimension

- Some columns may not fit to any of the dimensions, but creating a separate dimension for each of them would increase significantly the size of fact table (a foreign key would be required for each such column).
  - **Example**: consider such attributes as payment method (cash and credit card) and packing type (box, paper, or none).

Junk dimension

- Some columns may not fit to any of the dimensions, but creating a separate dimension for each of them would increase significantly the size of fact table (a foreign key would be required for each such column).
  - **Example**: consider such attributes as payment method (cash and credit card) and packing type (box, paper, or none).
- A possible solution is to create a single **junk dimension** combining these columns together.

Junk dimension

- Some columns may not fit to any of the dimensions, but creating a separate dimension for each of them would increase significantly the size of fact table (a foreign key would be required for each such column).
  - **Example**: consider such attributes as payment method (cash and credit card) and packing type (box, paper, or none).
- A possible solution is to create a single **junk dimension** combining these columns together.
- This dimension does not need to be the Cartesian product of all the attributes' possible values, but should only contain the combination of values that actually occur in the source data.

Slowly changing dimensions

- Compared to **fact tables**, **dimension tables** are relatively **stable**:

Slowly changing dimensions

- Compared to **fact tables**, **dimension tables** are relatively **stable**:
  - New sales transactions occur constantly,

Slowly changing dimensions

- Compared to **fact tables**, **dimension tables** are relatively **stable**:
  - New sales transactions occur constantly,
  - New products are introduced rarely,

Slowly changing dimensions

- Compared to **fact tables**, **dimension tables** are relatively **stable**:
  - New sales transactions occur constantly,
  - New products are introduced rarely,
  - New stores are opened very rarely.

Slowly changing dimensions

- Compared to **fact tables**, **dimension tables** are relatively **stable**:
  - New sales transactions occur constantly,
  - New products are introduced rarely,
  - New stores are opened very rarely.
- Attribute values for existing dimension rows do occasionally **change over time**:

Slowly changing dimensions

- Compared to **fact tables**, **dimension tables** are relatively **stable**:
  - New sales transactions occur constantly,
  - New products are introduced rarely,
  - New stores are opened very rarely.
- Attribute values for existing dimension rows do occasionally **change over time**:
  - Customer moves to a new address,

Slowly changing dimensions

- Compared to **fact tables**, **dimension tables** are relatively **stable**:
  - New sales transactions occur constantly,
  - New products are introduced rarely,
  - New stores are opened very rarely.
- Attribute values for existing dimension rows do occasionally **change over time**:
  - Customer moves to a new address,
  - Administrative reform in Poland,

Slowly changing dimensions

- Compared to **fact tables**, **dimension tables** are relatively **stable**:
  - New sales transactions occur constantly,
  - New products are introduced rarely,
  - New stores are opened very rarely.
- Attribute values for existing dimension rows do occasionally **change over time**:
  - Customer moves to a new address,
  - Administrative reform in Poland,
  - Change of product categorization.

Slowly changing dimensions

- **Type I** (the simplest solution): update the dimension table:

Slowly changing dimensions

- **Type I** (the simplest solution): update the dimension table:
  - Simple to implement,

Slowly changing dimensions

- **Type I** (the simplest solution): update the dimension table:
  - Simple to implement,
  - Does not report history,

Slowly changing dimensions

- **Type I** (the simplest solution): update the dimension table:
  - Simple to implement,
  - Does not report history,
  - Can affect analysis.

Slowly changing dimensions

- **Type I** (the simplest solution): update the dimension table:
  - Simple to implement,
  - Does not report history,
  - Can affect analysis.
- **Examples**

Slowly changing dimensions

- **Type I** (the simplest solution): update the dimension table:
  - Simple to implement,
  - Does not report history,
  - Can affect analysis.
- **Examples**
  - Wrong name of the street (correct approach),

Slowly changing dimensions

- **Type I** (the simplest solution): update the dimension table:
  - Simple to implement,
  - Does not report history,
  - Can affect analysis.
- **Examples**
  - Wrong name of the street (correct approach),
  - Change of customer address:
    Nowak moves from Poznań to Warsaw $\rightarrow$ all products purchased in Poznań move to Warsaw too!!!

Slowly changing dimensions

- **Type II**: create a new dimension row:

Slowly changing dimensions

- **Type II**: create a new dimension row:
  - Accurate historical reporting,

Slowly changing dimensions

- **Type II**: create a new dimension row:
  - Accurate historical reporting,
  - Pre-computed aggregates unaffected,

Slowly changing dimensions

- **Type II**: create a new dimension row:
  - Accurate historical reporting,
  - Pre-computed aggregates unaffected,
  - Dimension table grows over time,

Slowly changing dimensions

- **Type II**: create a new dimension row:
  - Accurate historical reporting,
  - Pre-computed aggregates unaffected,
  - Dimension table grows over time,
  - Type II requires surrogate keys.

Slowly changing dimensions

- **Type II**: create a new dimension row:
  - Accurate historical reporting,
  - Pre-computed aggregates unaffected,
  - Dimension table grows over time,
  - Type II requires surrogate keys.
- **Examples**:

Slowly changing dimensions

- **Type II**: create a new dimension row:
  - Accurate historical reporting,
  - Pre-computed aggregates unaffected,
  - Dimension table grows over time,
  - Type II requires surrogate keys.
- **Examples**:
  - Nowak moves to Warsaw:

id	local_id	name	city	row effective date	row expiration date	current row indicator
23401	234	Nowak	Poznań	2012-01-01	2014-01-05	Expired
23402	234	Nowak	Warsaw	2014-01-06	9999-12-31	Current

Slowly changing dimensions

- **Type II**: create a new dimension row:
  - Accurate historical reporting,
  - Pre-computed aggregates unaffected,
  - Dimension table grows over time,
  - Type II requires surrogate keys.
- **Examples**:
  - Nowak moves to Warsaw:

id	local_id	name	city	row effective date	row expiration date	current row indicator
23401	234	Nowak	Poznań	2012-01-01	2014-01-05	Expired
23402	234	Nowak	Warsaw	2014-01-06	9999-12-31	Current

  - Old fact table rows point to the old row, and new fact table rows point to the new row,

Slowly changing dimensions

- **Type II**: create a new dimension row:
  - Accurate historical reporting,
  - Pre-computed aggregates unaffected,
  - Dimension table grows over time,
  - Type II requires surrogate keys.
- **Examples**:
  - Nowak moves to Warsaw:

id	local_id	name	city	row effective date	row expiration date	current row indicator
23401	234	Nowak	Poznań	2012-01-01	2014-01-05	Expired
23402	234	Nowak	Warsaw	2014-01-06	9999-12-31	Current

  - Old fact table rows point to the old row, and new fact table rows point to the new row,
  - To report on Nowak's activity over time one can use:

```
WHERE local_id = "234"
```

Slowly changing dimensions

- **Type III**: create a new dimension column:

Slowly changing dimensions

- **Type III**: create a new dimension column:
  - Allows reporting using either the old or the new values,

Slowly changing dimensions

- **Type III**: create a new dimension column:
  - Allows reporting using either the old or the new values,
  - Mostly useful for different reorganizations.

Slowly changing dimensions

- **Type III**: create a new dimension column:
  - Allows reporting using either the old or the new values,
  - Mostly useful for different reorganizations.
- **Examples**:

Slowly changing dimensions

- **Type III**: create a new dimension column:
  - Allows reporting using either the old or the new values,
  - Mostly useful for different reorganizations.
- **Examples**:
  - Administrative reform in Poland:

id	previous name	current name
23401	poznańskie	wielkopolskie
23402	pilskie	wielkopolskie

Slowly changing dimensions

- **Type III**: create a new dimension column:
  - Allows reporting using either the old or the new values,
  - Mostly useful for different reorganizations.
- **Examples**:
  - Administrative reform in Poland:

id	previous name	current name
23401	poznańskie	wielkopolskie
23402	pilskie	wielkopolskie

  - All the fact table rows point to both values.

Mini dimension

- Some attributes change or are queried frequently.

Mini dimension

- Some attributes change or are queried frequently.
- For such attributes it may be reasonable to create a separate dimension table called **mini dimension**.

Mini dimension

- Some attributes change or are queried frequently.
- For such attributes it may be reasonable to create a separate dimension table called **mini dimension**.
- It relies on removing of frequently-changing or frequently-queried attributes from an original dimension and adding them to a new mini-dimension table.

Mini dimension

- Some attributes change or are queried frequently.
- For such attributes it may be reasonable to create a separate dimension table called **mini dimension**.
- It relies on removing of frequently-changing or frequently-queried attributes from an original dimension and adding them to a new mini-dimension table.
- The link between the original dimension and the mini dimension is through the fact table, but additional foreign key can be added to the original dimension.

Factless fact tables

- Fact tables take the advantage of sparsity (much less data to store, if events are rare).

Factless fact tables

- Fact tables take the advantage of sparsity (much less data to store, if events are rare).
- Fact tables do not contain rows for non-events: no rows for products that did not sell.

Factless fact tables

- Fact tables take the advantage of sparsity (much less data to store, if events are rare).
- Fact tables do not contain rows for non-events: no rows for products that did not sell.
- Factless Fact Tables:

Factless fact tables

- Fact tables take the advantage of sparsity (much less data to store, if events are rare).
- Fact tables do not contain rows for non-events: no rows for products that did not sell.
- Factless Fact Tables:
  - Fact table without numeric measure columns.

Factless fact tables

- Fact tables take the advantage of sparsity (much less data to store, if events are rare).
- Fact tables do not contain rows for non-events: no rows for products that did not sell.
- Factless Fact Tables:
  - Fact table without numeric measure columns.
  - Dummy measure is included that always has value 1.

Factless fact tables

- Fact tables take the advantage of sparsity (much less data to store, if events are rare).
- Fact tables do not contain rows for non-events: no rows for products that did not sell.
- Factless Fact Tables:
  - Fact table without numeric measure columns.
  - Dummy measure is included that always has value 1.
  - Describe relationships between dimensions.

Factless fact tables

- Fact tables take the advantage of sparsity (much less data to store, if events are rare).
- Fact tables do not contain rows for non-events: no rows for products that did not sell.
- Factless Fact Tables:
  - Fact table without numeric measure columns.
  - Dummy measure is included that always has value 1.
  - Describe relationships between dimensions.
  - **Example**: Which products were on promotion in which stores for which days?

Data warehouse bus architecture

- In order to create an data warehouse for entire organization, which takes into account several processes, one can use a **data warehouse bus matrix**.

Business process	Common dimensions	Date	Product	Store	Promotion	Warehouse	Vendor	Contract	Shipper
Retail sales		*	*	*	*				
Retail inventory		*	*	*					
Retail deliveries		*	*	*					
Warehouse inventory		*	*			*	*		
Warehouse deliveries		*	*			*	*		
Purchase orders		*	*			*	*	*	*
...									

Some other problems

- Querying fact tables or dimensions?

  It is generally reckoned that some 80 percent of data warehouse queries are dimension-browsing queries. This means that they do not access any fact table.

  Ch. Todman (2001)

- **Example**:
  - How many customers do we have?

Outline

Summary

- Three goals of the logical design of data warehouse: **simplicity**, **expressiveness** and **performance**.
- The most popular conceptual schema: **star schema**.
- Designing data warehouses is not an easy task ...

Bibliography

- Ch. Todman. *Designing a Data Warehouse: Supporting Customer Relationship Management*.
  Prentice Hall PTR, 2001
- Z. Królikowski. *Hurtownie danych: logiczne i fizyczne struktury danych*.
  Wydawnictwo Politechniki Poznańskiej, 2007
- R. Kimball and M. Ross. *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling, 3rd Edition*.
  John Wiley & Sons, 2013
- `http://www.kimballgroup.com/`