ETL and OLAP Systems

Krzysztof Dembczyński

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

Intelligent Decision Support Systems
Master studies, second semester
Academic year 2017/18 (summer course)

Review of the Previous Lecture

- Mining of massive datasets.
- Evolution of database systems.
- Dimensional modeling:
  - 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 . . .

Outline

Outline

Motivation

- OLAP queries are usually performed in a separate system, i.e., a data warehouse.

Motivation

- OLAP queries are usually performed in a separate system, i.e., a data warehouse.
- Transferring data to data warehouse:

Motivation

- OLAP queries are usually performed in a separate system, i.e., a data warehouse.
- Transferring data to data warehouse:
  - Data warehouses combine data from multiple sources.

Motivation

- OLAP queries are usually performed in a separate system, i.e., a data warehouse.
- Transferring data to data warehouse:
  - Data warehouses combine data from multiple sources.
  - Data must be translated into a consistent format.

Motivation

- OLAP queries are usually performed in a separate system, i.e., a data warehouse.
- Transferring data to data warehouse:
  - Data warehouses combine data from multiple sources.
  - Data must be translated into a consistent format.
  - Data integration represents 80% of effort for a typical data warehouse project!

Motivation

- OLAP queries are usually performed in a separate system, i.e., a data warehouse.
- Transferring data to data warehouse:
  - Data warehouses combine data from multiple sources.
  - Data must be translated into a consistent format.
  - Data integration represents 80% of effort for a typical data warehouse project!
- Optimization of data warehouse:

Motivation

- OLAP queries are usually performed in a separate system, i.e., a data warehouse.
- Transferring data to data warehouse:
  - Data warehouses combine data from multiple sources.
  - Data must be translated into a consistent format.
  - Data integration represents 80% of effort for a typical data warehouse project!
- Optimization of data warehouse:
  - Data storage: relational or multi-dimensional.

Motivation

- OLAP queries are usually performed in a separate system, i.e., a data warehouse.
- Transferring data to data warehouse:
  - Data warehouses combine data from multiple sources.
  - Data must be translated into a consistent format.
  - Data integration represents 80% of effort for a typical data warehouse project!
- Optimization of data warehouse:
  - Data storage: relational or multi-dimensional.
  - Additional data structures: sorting, indexing, summarizing, cubes.

Motivation

- OLAP queries are usually performed in a separate system, i.e., a data warehouse.
- Transferring data to data warehouse:
  - Data warehouses combine data from multiple sources.
  - Data must be translated into a consistent format.
  - Data integration represents 80% of effort for a typical data warehouse project!
- Optimization of data warehouse:
  - Data storage: relational or multi-dimensional.
  - Additional data structures: sorting, indexing, summarizing, cubes.
  - Refreshing of data structures.

Motivation

- OLAP queries are usually performed in a separate system, i.e., a data warehouse.
- Transferring data to data warehouse:
  - Data warehouses combine data from multiple sources.
  - Data must be translated into a consistent format.
  - Data integration represents 80% of effort for a typical data warehouse project!
- Optimization of data warehouse:
  - Data storage: relational or multi-dimensional.
  - Additional data structures: sorting, indexing, summarizing, cubes.
  - Refreshing of data structures.
- Querying multidimensional data:

Motivation

- OLAP queries are usually performed in a separate system, i.e., a data warehouse.
- Transferring data to data warehouse:
  - Data warehouses combine data from multiple sources.
  - Data must be translated into a consistent format.
  - Data integration represents 80% of effort for a typical data warehouse project!
- Optimization of data warehouse:
  - Data storage: relational or multi-dimensional.
  - Additional data structures: sorting, indexing, summarizing, cubes.
  - Refreshing of data structures.
- Querying multidimensional data:
  - SQL extensions,

Motivation

- OLAP queries are usually performed in a separate system, i.e., a data warehouse.
- Transferring data to data warehouse:
  - Data warehouses combine data from multiple sources.
  - Data must be translated into a consistent format.
  - Data integration represents 80% of effort for a typical data warehouse project!
- Optimization of data warehouse:
  - Data storage: relational or multi-dimensional.
  - Additional data structures: sorting, indexing, summarizing, cubes.
  - Refreshing of data structures.
- Querying multidimensional data:
  - SQL extensions,
  - Multidimensional expressions (MDX),

Motivation

- OLAP queries are usually performed in a separate system, i.e., a data warehouse.
- Transferring data to data warehouse:
  - Data warehouses combine data from multiple sources.
  - Data must be translated into a consistent format.
  - Data integration represents 80% of effort for a typical data warehouse project!
- Optimization of data warehouse:
  - Data storage: relational or multi-dimensional.
  - Additional data structures: sorting, indexing, summarizing, cubes.
  - Refreshing of data structures.
- Querying multidimensional data:
  - SQL extensions,
  - Multidimensional expressions (MDX),
  - Map-reduce-based languages.

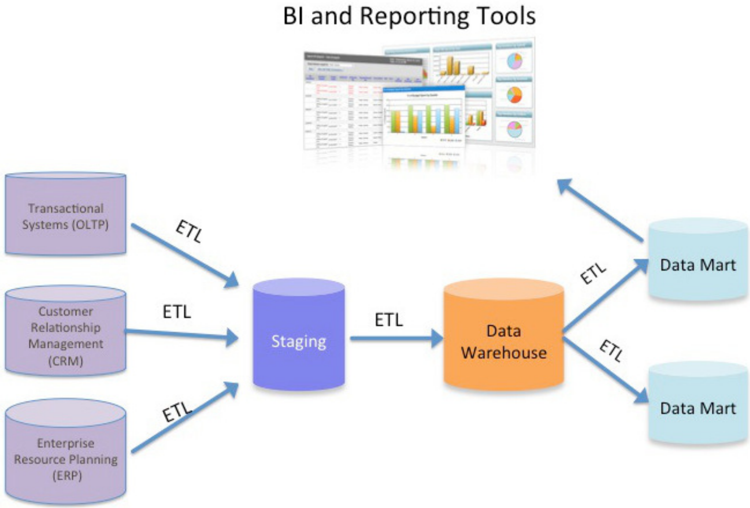
Outline

ETL

- **ETL** = Extraction, Transformation, and Load
  - **Extraction** of data from source systems,
  - **Transformation** and **integration** of data into a useful format for analysis,
  - **Load** of data into the warehouse and build of additional structures.
- **Refreshment** of data warehouse is closely related to ETL process.
- The ETL process is described by metadata stored in data warehouse.
- Architecture of data warehousing:

  Data sources $\Rightarrow$ Data staging area $\Rightarrow$ Data warehouse

ETL



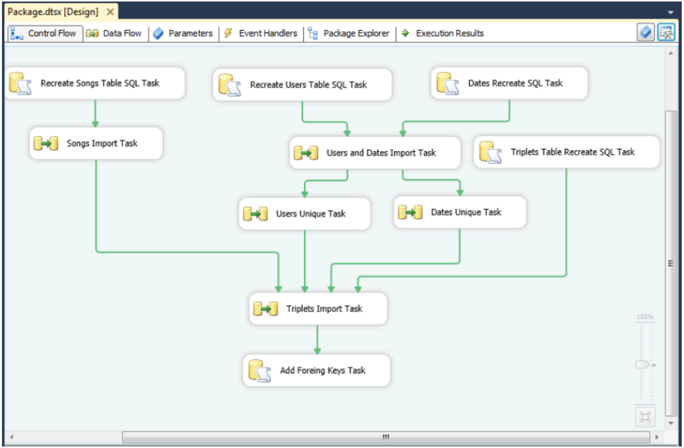
BI and Reporting Tools
Transactional Systems (OLTP)
Customer Relationship Management (CRM)
Enterprise Resource Planning (ERP)
ETL
ETL
ETL
Staging
ETL
Data Warehouse
ETL
ETL
Data Mart
Data Mart

Tools for ETL

- Data extraction from heterogeneous data sources.
- Data transformation, integration, and cleansing.
- Data quality analysis and control.
- Data loading.
- High-speed data transfer.
- Data refreshment.
- Managing and analyzing metadata.
- **Examples of ETL tools**:
  - MS SQL Server Integration Services(SSIS), IBM Infosphere DataStage, SAS ETL Studio, Oracle Warehouse Builder, Oracle Data Integrator, Business Objects Data Integrator, Pentaho Data Integration.

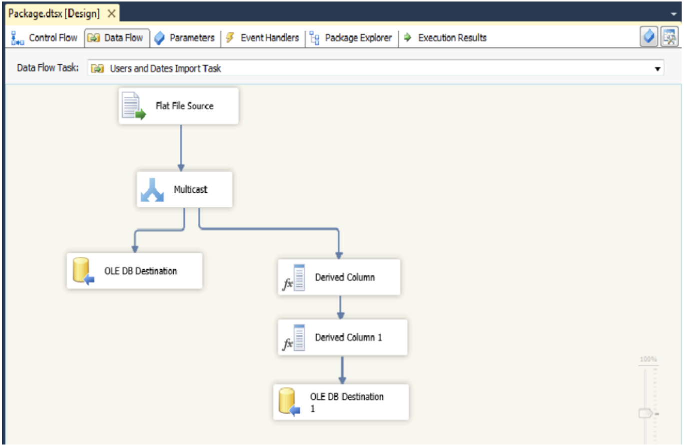
Tools for ETL

- MS SQL Server Integration Services(SSIS)



Tools for ETL

- MS SQL Server Integration Services(SSIS)



Data extraction

- Data warehouse needs extraction of data from different external data sources:

Data extraction

- Data warehouse needs extraction of data from different external data sources:
  - operational databases (relational, hierarchical, network, itp.),

Data extraction

- Data warehouse needs extraction of data from different external data sources:
  - operational databases (relational, hierarchical, network, itp.),
  - files of standard applications (Excel, COBOL applications),

Data extraction

- Data warehouse needs extraction of data from different external data sources:
  - operational databases (relational, hierarchical, network, itp.),
  - files of standard applications (Excel, COBOL applications),
  - additional databases (direct marketing databases) and data services (stock data),

Data extraction

- Data warehouse needs extraction of data from different external data sources:
  - operational databases (relational, hierarchical, network, itp.),
  - files of standard applications (Excel, COBOL applications),
  - additional databases (direct marketing databases) and data services (stock data),
  - various log files,

Data extraction

- Data warehouse needs extraction of data from different external data sources:
  - operational databases (relational, hierarchical, network, itp.),
  - files of standard applications (Excel, COBOL applications),
  - additional databases (direct marketing databases) and data services (stock data),
  - various log files,
  - and other documents (.txt, .doc, XML, WWW).

Data extraction

- Data warehouse needs extraction of data from different external data sources:
  - operational databases (relational, hierarchical, network, itp.),
  - files of standard applications (Excel, COBOL applications),
  - additional databases (direct marketing databases) and data services (stock data),
  - various log files,
  - and other documents (.txt, .doc, XML, WWW).
- Access to data sources can be difficult:

Data extraction

- Data warehouse needs extraction of data from different external data sources:
  - operational databases (relational, hierarchical, network, itp.),
  - files of standard applications (Excel, COBOL applications),
  - additional databases (direct marketing databases) and data services (stock data),
  - various log files,
  - and other documents (.txt, .doc, XML, WWW).
- Access to data sources can be difficult:
  - Data sources are often operational systems, providing the lowest level of data.

Data extraction

- Data warehouse needs extraction of data from different external data sources:
  - operational databases (relational, hierarchical, network, itp.),
  - files of standard applications (Excel, COBOL applications),
  - additional databases (direct marketing databases) and data services (stock data),
  - various log files,
  - and other documents (.txt, .doc, XML, WWW).
- Access to data sources can be difficult:
  - Data sources are often operational systems, providing the lowest level of data.
  - Data sources are designed for operational use, not for decision support, and the data reflect this fact.

Data extraction

- Data warehouse needs extraction of data from different external data sources:
  - operational databases (relational, hierarchical, network, itp.),
  - files of standard applications (Excel, COBOL applications),
  - additional databases (direct marketing databases) and data services (stock data),
  - various log files,
  - and other documents (.txt, .doc, XML, WWW).
- Access to data sources can be difficult:
  - Data sources are often operational systems, providing the lowest level of data.
  - Data sources are designed for operational use, not for decision support, and the data reflect this fact.
  - Multiple data sources are often from different systems, run on a wide range of hardware and much of the software is built in-house or highly customized.

Data extraction

- Data warehouse needs extraction of data from different external data sources:
  - operational databases (relational, hierarchical, network, itp.),
  - files of standard applications (Excel, COBOL applications),
  - additional databases (direct marketing databases) and data services (stock data),
  - various log files,
  - and other documents (.txt, .doc, XML, WWW).
- Access to data sources can be difficult:
  - Data sources are often operational systems, providing the lowest level of data.
  - Data sources are designed for operational use, not for decision support, and the data reflect this fact.
  - Multiple data sources are often from different systems, run on a wide range of hardware and much of the software is built in-house or highly customized.
  - Data sources can be designed using different logical structures.

Data extraction

- Identification of concepts and objects does not have to be easy.

Data extraction

- Identification of concepts and objects does not have to be easy.
- **Example**: Extract information about sales from the source system.

Data extraction

- Identification of concepts and objects does not have to be easy.
- **Example**: Extract information about sales from the source system.
    - What is meant by the term **sale**? A sale has occurred when

Data extraction

- Identification of concepts and objects does not have to be easy.
- **Example**: Extract information about sales from the source system.
  - What is meant by the term **sale**? A sale has occurred when
    1. the order has been received by a customer,

Data extraction

- Identification of concepts and objects does not have to be easy.
- **Example**: Extract information about sales from the source system.
  - What is meant by the term **sale**? A sale has occurred when
    1. the order has been received by a customer,
    2. the order is sent to the customer,

Data extraction

- Identification of concepts and objects does not have to be easy.
- **Example**: Extract information about sales from the source system.
  - What is meant by the term **sale**? A sale has occurred when
    1. the order has been received by a customer,
    2. the order is sent to the customer,
    3. the invoice has been raised against the order.

Data extraction

- Identification of concepts and objects does not have to be easy.
- **Example**: Extract information about sales from the source system.
  - What is meant by the term **sale**? A sale has occurred when
    1. the order has been received by a customer,
    2. the order is sent to the customer,
    3. the invoice has been raised against the order.
  - It is a common problem that there is no table `SALES` in the operational databases; some other tables can exist like `ORDER` with an attribute `ORDER_STATUS`.

Conflicts and dirty data

- Different logical models of operational sources,

Conflicts and dirty data

- Different logical models of operational sources,
- Different data types (account number stored as **String** or **Numeric**),

Conflicts and dirty data

- Different logical models of operational sources,
- Different data types (account number stored as **String** or **Numeric**),
- Different data domains (gender: **M**, **F**, **male**, **female**, 1, 0),

Conflicts and dirty data

- Different logical models of operational sources,
- Different data types (account number stored as **String** or **Numeric**),
- Different data domains (gender: **M**, **F**, **male**, **female**, 1, 0),
- Different date formats (dd-mm-yyyy or mm-dd-yyyy),

Conflicts and dirty data

- Different logical models of operational sources,
- Different data types (account number stored as **String** or **Numeric**),
- Different data domains (gender: **M**, **F**, **male**, **female**, 1, 0),
- Different date formats (dd-mm-yyyy or mm-dd-yyyy),
- Different field lengths (address stored by using 20 or 50 chars),

Conflicts and dirty data

- Different logical models of operational sources,
- Different data types (account number stored as **String** or **Numeric**),
- Different data domains (gender: **M**, **F**, **male**, **female**, 1, 0),
- Different date formats (dd-mm-yyyy or mm-dd-yyyy),
- Different field lengths (address stored by using 20 or 50 chars),
- Different naming conventions: homonyms and synonyms,

Conflicts and dirty data

- Different logical models of operational sources,
- Different data types (account number stored as **String** or **Numeric**),
- Different data domains (gender: **M**, **F**, **male**, **female**, 1, 0),
- Different date formats (dd-mm-yyyy or mm-dd-yyyy),
- Different field lengths (address stored by using 20 or 50 chars),
- Different naming conventions: homonyms and synonyms,
- Missing values and dirty data,

Conflicts and dirty data

- Different logical models of operational sources,
- Different data types (account number stored as **String** or **Numeric**),
- Different data domains (gender: **M**, **F**, **male**, **female**, 1, 0),
- Different date formats (dd-mm-yyyy or mm-dd-yyyy),
- Different field lengths (address stored by using 20 or 50 chars),
- Different naming conventions: homonyms and synonyms,
- Missing values and dirty data,
- Inconsistent information concerning the same object,

Conflicts and dirty data

- Different logical models of operational sources,
- Different data types (account number stored as **String** or **Numeric**),
- Different data domains (gender: **M**, **F**, **male**, **female**, 1, 0),
- Different date formats (dd-mm-yyyy or mm-dd-yyyy),
- Different field lengths (address stored by using 20 or 50 chars),
- Different naming conventions: homonyms and synonyms,
- Missing values and dirty data,
- Inconsistent information concerning the same object,
- Information concerning the same object, but indicated by different keys,

Conflicts and dirty data

- Different logical models of operational sources,
- Different data types (account number stored as **String** or **Numeric**),
- Different data domains (gender: **M**, **F**, **male**, **female**, 1, 0),
- Different date formats (dd-mm-yyyy or mm-dd-yyyy),
- Different field lengths (address stored by using 20 or 50 chars),
- Different naming conventions: homonyms and synonyms,
- Missing values and dirty data,
- Inconsistent information concerning the same object,
- Information concerning the same object, but indicated by different keys,
- ...

Deduplication and householding

- **Deduplication** ensures that one accurate record exists for each business entity represented in a database,

Deduplication and householding

- **Deduplication** ensures that one accurate record exists for each business entity represented in a database,
- **Householding** is the technique of grouping individual customers by the household or organization of which they are a member; this technique has some interesting marketing implications, and can also support cost-saving measures of direct advertising.

Deduplication and householding

- **Deduplication** ensures that one accurate record exists for each business entity represented in a database,
- **Householding** is the technique of grouping individual customers by the household or organization of which they are a member; this technique has some interesting marketing implications, and can also support cost-saving measures of direct advertising.
- **Example**:

Deduplication and householding

- **Deduplication** ensures that one accurate record exists for each business entity represented in a database,
- **Householding** is the technique of grouping individual customers by the household or organization of which they are a member; this technique has some interesting marketing implications, and can also support cost-saving measures of direct advertising.
- **Example**:
  - Consider the following rows in a database:

Tim Jones	123	Main Street	Marlboro	MA	12234
T. Jones	123	Main St.	Marlborogh	MA	12234
Timothy Jones	321	Maine Street	Marlborog	AM	12234
Jones, Timothy	123	Maine Ave	Marlborough	MA	13324

Deduplication and householding

- **Deduplication** ensures that one accurate record exists for each business entity represented in a database,
- **Householding** is the technique of grouping individual customers by the household or organization of which they are a member; this technique has some interesting marketing implications, and can also support cost-saving measures of direct advertising.
- **Example**:
  - Consider the following rows in a database:

Tim Jones	123	Main Street	Marlboro	MA	12234
T. Jones	123	Main St.	Marlborogh	MA	12234
Timothy Jones	321	Maine Street	Marlborog	AM	12234
Jones, Timothy	123	Maine Ave	Marlborough	MA	13324

  - The sales for around $500 are counted for each tuple.

Deduplication and householding

- **Deduplication** ensures that one accurate record exists for each business entity represented in a database,
- **Householding** is the technique of grouping individual customers by the household or organization of which they are a member; this technique has some interesting marketing implications, and can also support cost-saving measures of direct advertising.
- **Example**:
  - Consider the following rows in a database:

Tim Jones	123	Main Street	Marlboro	MA	12234
T. Jones	123	Main St.	Marlborogh	MA	12234
Timothy Jones	321	Maine Street	Marlborog	AM	12234
Jones, Timothy	123	Maine Ave	Marlborough	MA	13324

  - The sales for around $500 are counted for each tuple.
  - Is it the same person?

Load of data

- After extracting, cleaning and transforming, data must be loaded into the warehouse.

Load of data

- After extracting, cleaning and transforming, data must be loaded into the warehouse.
- Loading the warehouse includes some other processing tasks: checking integrity constraints, sorting, summarizing, creating indexes, etc.

Load of data

- After extracting, cleaning and transforming, data must be loaded into the warehouse.
- Loading the warehouse includes some other processing tasks: checking integrity constraints, sorting, summarizing, creating indexes, etc.
- Batch (bulk) load utilities are used for loading.

Load of data

- After extracting, cleaning and transforming, data must be loaded into the warehouse.
- Loading the warehouse includes some other processing tasks: checking integrity constraints, sorting, summarizing, creating indexes, etc.
- Batch (bulk) load utilities are used for loading.
- A load utility must allow the administrator to monitor status, to cancel, suspend, and resume a load, and to restart after failure with no loss of data integrity.

Data warehouse refreshment

- Refreshing a warehouse means propagating updates on source data to the data stored in the warehouse.

Data warehouse refreshment

- Refreshing a warehouse means propagating updates on source data to the data stored in the warehouse.
- Follows the same structure as ETL process.

Data warehouse refreshment

- Refreshing a warehouse means propagating updates on source data to the data stored in the warehouse.
- Follows the same structure as ETL process.
- Several constraints: accessibility of data sources, size of data, size of data warehouse, frequency of data refreshing, degradation of performance of operational systems.

Data warehouse refreshment

- Refreshing a warehouse means propagating updates on source data to the data stored in the warehouse.
- Follows the same structure as ETL process.
- Several constraints: accessibility of data sources, size of data, size of data warehouse, frequency of data refreshing, degradation of performance of operational systems.
- Types of refreshments:

Data warehouse refreshment

- Refreshing a warehouse means propagating updates on source data to the data stored in the warehouse.
- Follows the same structure as ETL process.
- Several constraints: accessibility of data sources, size of data, size of data warehouse, frequency of data refreshing, degradation of performance of operational systems.
- Types of refreshments:
  - Periodical refreshment (daily or weekly).

Data warehouse refreshment

- Refreshing a warehouse means propagating updates on source data to the data stored in the warehouse.
- Follows the same structure as ETL process.
- Several constraints: accessibility of data sources, size of data, size of data warehouse, frequency of data refreshing, degradation of performance of operational systems.
- Types of refreshments:
  - Periodical refreshment (daily or weekly).
  - Immediate refreshment.

Data warehouse refreshment

- Refreshing a warehouse means propagating updates on source data to the data stored in the warehouse.
- Follows the same structure as ETL process.
- Several constraints: accessibility of data sources, size of data, size of data warehouse, frequency of data refreshing, degradation of performance of operational systems.
- Types of refreshments:
  - Periodical refreshment (daily or weekly).
  - Immediate refreshment.
  - Determined by usage, types of data source, etc.

Data warehouse refreshment

- Detect changes in external data sources:

Data warehouse refreshment

- Detect changes in external data sources:
  - Different monitoring techniques: external and intrusive techniques.

Data warehouse refreshment

- Detect changes in external data sources:
  - Different monitoring techniques: external and intrusive techniques.
  - Snapshot vs. timestamped sources

Data warehouse refreshment

- Detect changes in external data sources:
  - Different monitoring techniques: external and intrusive techniques.
  - Snapshot vs. timestamped sources
  - Queryable, logged, and replicated sources

Data warehouse refreshment

- Detect changes in external data sources:
  - Different monitoring techniques: external and intrusive techniques.
  - Snapshot vs. timestamped sources
  - Queryable, logged, and replicated sources
  - Callback and internal action sources

Data warehouse refreshment

- Detect changes in external data sources:
  - Different monitoring techniques: external and intrusive techniques.
  - Snapshot vs. timestamped sources
  - Queryable, logged, and replicated sources
  - Callback and internal action sources
- Extract the changes and integrate into the warehouse.

Data warehouse refreshment

- Detect changes in external data sources:
  - Different monitoring techniques: external and intrusive techniques.
  - Snapshot vs. timestamped sources
  - Queryable, logged, and replicated sources
  - Callback and internal action sources
- Extract the changes and integrate into the warehouse.
- Update indexes, subaggregates and any other additional data structures.

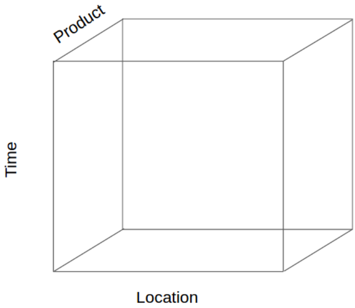
Outline

OLAP systems

- The next step is to provide solutions for querying and reporting multidimensional analytical data.

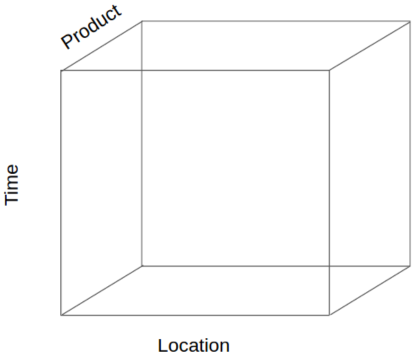
Multidimensional cube

- The proper data model for multidimensional reporting is the multidimensional one.



Operations in multidimensional data model

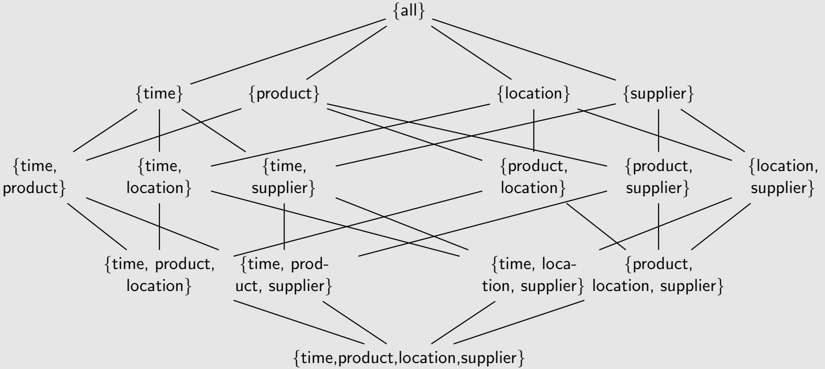
- Roll up – summarize data along a dimension hierarchy.
- Drill down – go from higher level summary to lower level summary or detailed data.
- Slice and dice – corresponds to selection and projection.
- Pivot – reorient cube.
- Raking, Time functions, etc.



Lattice of cuboids

- Different degrees of summarizations are presented as a lattice of cuboids.

Example for dimensions: time, product, location, supplier



Using this structure, one can easily show roll up and drill down operations.

Total number of cuboids

- For an n-dimensional data cube, the total number of cuboids that can be generated is:

$$T = \prod_{i=1}^{n} (L_i + 1)\,,$$

where L_i is the number of levels associated with dimension i (excluding the virtual top level "all" since generalizing to "all" is equivalent to the removal of a dimension).

- For example, if the cube has 10 dimensions and each dimension has 4 levels, the total number of cuboids that can be generated will be:

$$T = 5^{10} = 9,8 \times 10^6\,.$$

Total number of cuboids

- **Example**: Consider a simple database with two dimensions:

Total number of cuboids

- **Example**: Consider a simple database with two dimensions:
  - Columns in `Date` dimension: `day`, `month`, `year`
  - Columns in `Localization` dimension: `street`, `city`, `country`.
  - Without any information about hierarchies, the number of all possible group-bys is

Total number of cuboids

- **Example**: Consider a simple database with two dimensions:
  - Columns in `Date` dimension: `day`, `month`, `year`
  - Columns in `Localization` dimension: `street`, `city`, `country`.
  - Without any information about hierarchies, the number of all possible group-bys is 2^6:

Total number of cuboids

- **Example**: Consider a simple database with two dimensions:
  - Columns in `Date` dimension: `day`, `month`, `year`
  - Columns in `Localization` dimension: `street`, `city`, `country`.
  - Without any information about hierarchies, the number of all possible group-bys is 2^6:

$\emptyset$		$\emptyset$
day		street
month		city
year		country
day, month	$\bowtie$	street, city
day, year		street, country
month, year		city, country
day, month, year		street, city, country

Total number of cuboids

- **Example**: Consider the same relations but with defined hierarchies:

Total number of cuboids

- **Example**: Consider the same relations but with defined hierarchies:
  - `day` $\rightarrow$ `month` $\rightarrow$ year
  - `street` $\rightarrow$ `city` $\rightarrow$ `country`

Total number of cuboids

- **Example**: Consider the same relations but with defined hierarchies:
  - `day` $\rightarrow$ `month` $\rightarrow$ year
  - `street` $\rightarrow$ `city` $\rightarrow$ `country`
  - Many combinations of columns can be excluded, e.g., `group by day, year, street, country`.
  - The number of group-bys is then

Total number of cuboids

- **Example**: Consider the same relations but with defined hierarchies:
  - `day` $\rightarrow$ `month` $\rightarrow$ year
  - `street` $\rightarrow$ `city` $\rightarrow$ `country`
  - Many combinations of columns can be excluded, e.g., `group by day, year, street, country`.
  - The number of group-bys is then 4^2:

Total number of cuboids

- **Example**: Consider the same relations but with defined hierarchies:
  - `day` → `month` → year
  - `street` → `city` → `country`
  - Many combinations of columns can be excluded, e.g., `group by day, year, street, country`.
  - The number of group-bys is then 4^2:

$\emptyset$		$\emptyset$
year	$\bowtie$	country
month, year		city, country
day, month, year		street, city, country

Three types of aggregate functions

- distributive: `count()`, `sum()`, `max()`, `min()`,
- algebraic: `ave()`, `stddev()`,
- holistic: `median()`, `mode()`, `rank()`.

OLAP servers

- Relational OLAP (ROLAP),
- Multidimensional OLAP (MOLAP),
- Hybrid OLAP (HOLAP).

ROLAP

- **ROLAP servers** use a relational or post-relational database management system to store and manage warehouse data.

ROLAP

- **ROLAP servers** use a relational or post-relational database management system to store and manage warehouse data.
- ROLAP systems use SQL and its OLAP extensions.

ROLAP

- **ROLAP servers** use a relational or post-relational database management system to store and manage warehouse data.
- ROLAP systems use SQL and its OLAP extensions.
- Optimization techniques:

ROLAP

- **ROLAP servers** use a relational or post-relational database management system to store and manage warehouse data.
- ROLAP systems use SQL and its OLAP extensions.
- Optimization techniques:
  - Denormalization,

ROLAP

- **ROLAP servers** use a relational or post-relational database management system to store and manage warehouse data.
- ROLAP systems use SQL and its OLAP extensions.
- Optimization techniques:
  - Denormalization,
  - Materialized views,

ROLAP

- **ROLAP servers** use a relational or post-relational database management system to store and manage warehouse data.
- ROLAP systems use SQL and its OLAP extensions.
- Optimization techniques:
  - Denormalization,
  - Materialized views,
  - Partitioning,

ROLAP

- **ROLAP servers** use a relational or post-relational database management system to store and manage warehouse data.
- ROLAP systems use SQL and its OLAP extensions.
- Optimization techniques:
  - Denormalization,
  - Materialized views,
  - Partitioning,
  - Joins,

ROLAP

- **ROLAP servers** use a relational or post-relational database management system to store and manage warehouse data.
- ROLAP systems use SQL and its OLAP extensions.
- Optimization techniques:
  - Denormalization,
  - Materialized views,
  - Partitioning,
  - Joins,
  - Indexes (join index, bitmaps),

ROLAP

- **ROLAP servers** use a relational or post-relational database management system to store and manage warehouse data.
- ROLAP systems use SQL and its OLAP extensions.
- Optimization techniques:
  - Denormalization,
  - Materialized views,
  - Partitioning,
  - Joins,
  - Indexes (join index, bitmaps),
  - Query processing.

ROLAP

- Advantages of ROLAP Servers:

ROLAP

- Advantages of ROLAP Servers:
  - Scalable with respect to the number of dimensions,

ROLAP

- Advantages of ROLAP Servers:
  - Scalable with respect to the number of dimensions,
  - Scalable with respect to the size of data,

ROLAP

- Advantages of ROLAP Servers:
  - Scalable with respect to the number of dimensions,
  - Scalable with respect to the size of data,
  - Sparsity is not a problem (fact tables contain only facts),

ROLAP

- Advantages of ROLAP Servers:
  - Scalable with respect to the number of dimensions,
  - Scalable with respect to the size of data,
  - Sparsity is not a problem (fact tables contain only facts),
  - Mature and well-developed technology.

ROLAP

- Advantages of ROLAP Servers:
  - Scalable with respect to the number of dimensions,
  - Scalable with respect to the size of data,
  - Sparsity is not a problem (fact tables contain only facts),
  - Mature and well-developed technology.
- Disadvantage of ROLAP Servers:

ROLAP

- Advantages of ROLAP Servers:
  - Scalable with respect to the number of dimensions,
  - Scalable with respect to the size of data,
  - Sparsity is not a problem (fact tables contain only facts),
  - Mature and well-developed technology.
- Disadvantage of ROLAP Servers:
  - Worse performance than MOLAP,

ROLAP

- Advantages of ROLAP Servers:
  - Scalable with respect to the number of dimensions,
  - Scalable with respect to the size of data,
  - Sparsity is not a problem (fact tables contain only facts),
  - Mature and well-developed technology.
- Disadvantage of ROLAP Servers:
  - Worse performance than MOLAP,
  - Additional data structures and optimization techniques used to improve the performance.

MOLAP

- MOLAP Servers use array-based multidimensional storage engines.

MOLAP

- MOLAP Servers use array-based multidimensional storage engines.
- Optimization techniques:

MOLAP

- MOLAP Servers use array-based multidimensional storage engines.
- Optimization techniques:
  - Two-level storage representation: dense cubes are identified and stored as array structures, sparse cubes employ compression techniques,

MOLAP

- MOLAP Servers use array-based multidimensional storage engines.
- Optimization techniques:
  - Two-level storage representation: dense cubes are identified and stored as array structures, sparse cubes employ compression techniques,
  - Materialized cubes.

MOLAP

- Advantages of MOLAP Servers:

MOLAP

- Advantages of MOLAP Servers:
  - Multidimensional views are directly mapped to data cube array structures – efficient access to data,

MOLAP

- Advantages of MOLAP Servers:
  - Multidimensional views are directly mapped to data cube array structures – efficient access to data,
  - Can easily store subaggregates.

MOLAP

- Advantages of MOLAP Servers:
  - Multidimensional views are directly mapped to data cube array structures – efficient access to data,
  - Can easily store subaggregates.
- Disadvantages of MOLAP Servers:

MOLAP

- Advantages of MOLAP Servers:
  - Multidimensional views are directly mapped to data cube array structures – efficient access to data,
  - Can easily store subaggregates.
- Disadvantages of MOLAP Servers:
  - Scalability problem in the case of larger number of dimensions,

MOLAP

- Advantages of MOLAP Servers:
  - Multidimensional views are directly mapped to data cube array structures – efficient access to data,
  - Can easily store subaggregates.
- Disadvantages of MOLAP Servers:
  - Scalability problem in the case of larger number of dimensions,
  - Not tailored for sparse data.

MOLAP

- Advantages of MOLAP Servers:
  - Multidimensional views are directly mapped to data cube array structures – efficient access to data,
  - Can easily store subaggregates.
- Disadvantages of MOLAP Servers:
  - Scalability problem in the case of larger number of dimensions,
  - Not tailored for sparse data.
  - **Example**:

MOLAP

- Advantages of MOLAP Servers:
  - Multidimensional views are directly mapped to data cube array structures – efficient access to data,
  - Can easily store subaggregates.
- Disadvantages of MOLAP Servers:
  - Scalability problem in the case of larger number of dimensions,
  - Not tailored for sparse data.
  - **Example**:
    - Logical model consists of four dimensions: customer, product, location, and day

MOLAP

- Advantages of MOLAP Servers:
  - Multidimensional views are directly mapped to data cube array structures – efficient access to data,
  - Can easily store subaggregates.
- Disadvantages of MOLAP Servers:
  - Scalability problem in the case of larger number of dimensions,
  - Not tailored for sparse data.
  - **Example**:
    - Logical model consists of four dimensions: customer, product, location, and day
    - In case of 100 000 customers, 10 000 products, 1 000 locations and 1 000 days, the data cube will contain 1 000 000 000 000 000 cells!

MOLAP

- Advantages of MOLAP Servers:
  - Multidimensional views are directly mapped to data cube array structures – efficient access to data,
  - Can easily store subaggregates.
- Disadvantages of MOLAP Servers:
  - Scalability problem in the case of larger number of dimensions,
  - Not tailored for sparse data.
  - **Example**:
    - Logical model consists of four dimensions: customer, product, location, and day
    - In case of 100 000 customers, 10 000 products, 1 000 locations and 1 000 days, the data cube will contain 1 000 000 000 000 000 cells!
    - A huge number of cells is empty: a customer is not able to buy all products in all locations ...

HOLAP

- HOLAP servers are a hybrid approach that combines ROLAP and MOLAP technology.
- HOLAP benefits from the greater scalability of ROLAP and the faster computation of MOLAP.

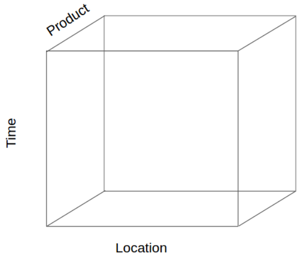
Outline

Multidimensional queries

- We need an intuitive way of expressing analytical (multidimensional) queries:

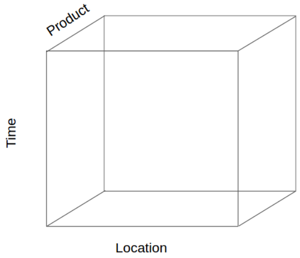
Multidimensional queries

- We need an intuitive way of expressing analytical (multidimensional) queries:
  - Operations like roll up, drill down, slice and dice, pivoting, ranking, time and window functions, etc.



Multidimensional queries

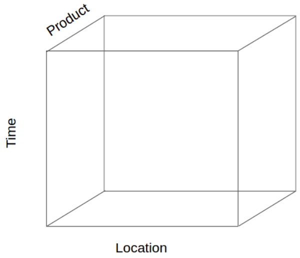
- We need an intuitive way of expressing analytical (multidimensional) queries:
  - Operations like roll up, drill down, slice and dice, pivoting, ranking, time and window functions, etc.



- Two solutions:

Multidimensional queries

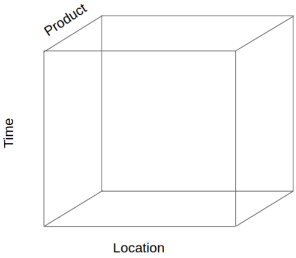
- We need an intuitive way of expressing analytical (multidimensional) queries:
  - Operations like roll up, drill down, slice and dice, pivoting, ranking, time and window functions, etc.



- Two solutions:
  - Extending **SQL**, or

Multidimensional queries

- We need an intuitive way of expressing analytical (multidimensional) queries:
  - Operations like roll up, drill down, slice and dice, pivoting, ranking, time and window functions, etc.



- Two solutions:
  - Extending **SQL**, or
  - Inventing a new language ($\rightarrow$ **MDX**).

OLAP queries in SQL

- A typical example of an analytical query is a group-by query:

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

- And the result:

Academic_year	Name	AVG(Grade)
2013/14	Stefanowski	4.2
2014/15	Stefanowski	4.5
2013/14	Słowiński	4.1
2014/15	Słowiński	4.3
2014/15	Dembczyński	4.6

SQL

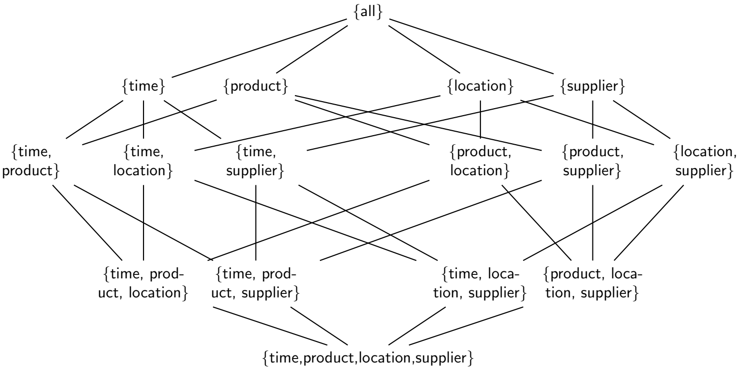
- OLAP extensions in SQL:
  - **GROUP BY CUBE**,
  - `GROUP BY ROLLUP`,
  - `GROUP BY GROUPING SETS`,
  - **OVER** and **PARTITION BY**,
  - `RANK`.

SQL

- `GROUP BY CUBE`

- GROUP BY CUBE
  - **Example**:

```
SELECT Time, Product, Location, Supplier, SUM(Gain)
FROM Sales
GROUP BY CUBE (Time, Product, Location, Supplier);
```



SQL

- GROUP BY CUBE
  - **Example**:

```
SELECT Time, Product, Location, Supplier, SUM(Gain)
FROM Sales
GROUP BY Time, Product, Location, Supplier
UNION ALL
SELECT Time, Product, Location, ''*'', SUM(Gain)
FROM Sales
GROUP BY Time, Product, Location
UNION ALL
SELECT Time, Product, ''*'', Location, SUM(Gain)
FROM Sales
GROUP BY Time, Product, Location
UNION ALL
...
UNION ALL
SELECT '*', '*', '*', '*', SUM(Gain)
FROM Sales;
```

SQL

- `GROUP BY CUBE`
  - It is not only a *Macro* instruction to reduce the number of subgroup-bys.

SQL

- `GROUP BY CUBE`
  - It is not only a *Macro* instruction to reduce the number of subgroup-bys.
  - One can easily optimize the group-by operations, when they are performed all-together: upper-level group-bys can be computed from lower-level group-bys.

SQL

- GROUP BY CUBE
  - **Example**:
    SELECT Academic year, Name, AVG(Grade) FROM Students_grades GROUP BY CUBE(Academic year, Name);

All rows and columns

Academic_year	Name	AVG(Grade)
2011/2	Stefanowski	4.2
2011/2	Słowiński	4.1
2012/3	Stefanowski	4.0
2012/3	Słowiński	3.8
2013/4	Stefanowski	3.9
2013/4	Słowiński	3.6
2013/4	Dembczyński	4.8

Academic_year	AVG(Grade)
2011/2	4.15
2012/3	3.85
2013/4	3.8

Name	AVG(Grade)
Stefanowski	3.9
Słowiński	3.6
Dembczyński	4.8

AVG(Grade)
3.95

SQL

- `GROUP BY CUBE`
  - **Example**:
    `SELECT Academic year, Name, AVG(Grade) FROM Students_grades GROUP BY CUBE(Academic year, Name);`

Academic_year	Name	AVG(Grade)
2011/2	Stefanowski	4.2
2011/2	Słowiński	4.1
2012/3	Stefanowski	4.0
2012/3	Słowiński	3.8
2013/4	Stefanowski	3.9
2013/4	Słowiński	3.6
2013/4	Dembczyński	4.8
2011/2	NULL	4.15
2012/3	NULL	3.85
2013/4	NULL	3.8
NULL	Stefanowski	3.9
NULL	Słowiński	3.6
NULL	Dembczyński	4.8
NULL	NULL	3.95

SQL

- `OVER()`:

SQL

- `OVER()`:
  - Determines the partitioning and ordering of a rowset before the associated window function is applied.

SQL

- `OVER()`:
  - Determines the partitioning and ordering of a rowset before the associated window function is applied.
  - The `OVER` clause defines a window or user-specified set of rows within a query result set.

SQL

- `OVER()`:
  - Determines the partitioning and ordering of a rowset before the associated window function is applied.
  - The `OVER` clause defines a window or user-specified set of rows within a query result set.
  - A window function then computes a value for each row in the window.

SQL

- `OVER()`:
  - Determines the partitioning and ordering of a rowset before the associated window function is applied.
  - The `OVER` clause defines a window or user-specified set of rows within a query result set.
  - A window function then computes a value for each row in the window.
  - The `OVER` clause can be used with functions to compute aggregated values such as moving averages, cumulative aggregates, running totals, or a top N per group results.

SQL

- `OVER()`:
  - Determines the partitioning and ordering of a rowset before the associated window function is applied.
  - The `OVER` clause defines a window or user-specified set of rows within a query result set.
  - A window function then computes a value for each row in the window.
  - The `OVER` clause can be used with functions to compute aggregated values such as moving averages, cumulative aggregates, running totals, or a top N per group results.
  - Syntax:

```
OVER (
    [ <PARTITION BY clause> ]
    [ <ORDER BY clause> ]
    [ <ROW or RANGE clause> ]
)
```

SQL

- `OVER()`:

SQL

- `OVER()`:
  - `PARTITION BY`:

SQL

- `OVER()`:
  - `PARTITION BY`:
    - Divides the query result set into partitions. The window function is applied to each partition separately and computation restarts for each partition.

SQL

- `OVER()`:
  - `PARTITION BY`:
    - Divides the query result set into partitions. The window function is applied to each partition separately and computation restarts for each partition.
  - `ORDER BY`:

SQL

- `OVER()`:
  - `PARTITION BY`:
    - Divides the query result set into partitions. The window function is applied to each partition separately and computation restarts for each partition.
  - `ORDER BY`:
    - Defines the logical order of the rows within each partition of the result set, i.e., it specifies the logical order in which the window function calculation is performed.

SQL

- `OVER()`:
  - `PARTITION BY`:
    - Divides the query result set into partitions. The window function is applied to each partition separately and computation restarts for each partition.
  - `ORDER BY`:
    - Defines the logical order of the rows within each partition of the result set, i.e., it specifies the logical order in which the window function calculation is performed.
  - `ROW` and `RANGE`:

SQL

- `OVER()`:
  - `PARTITION BY`:
    - Divides the query result set into partitions. The window function is applied to each partition separately and computation restarts for each partition.
  - `ORDER BY`:
    - Defines the logical order of the rows within each partition of the result set, i.e., it specifies the logical order in which the window function calculation is performed.
  - `ROW` and `RANGE`:
    - Further limits the rows within the partition by specifying start and end points within the partition.

SQL

- `OVER()`:
  - `PARTITION BY`:
    - Divides the query result set into partitions. The window function is applied to each partition separately and computation restarts for each partition.
  - `ORDER BY`:
    - Defines the logical order of the rows within each partition of the result set, i.e., it specifies the logical order in which the window function calculation is performed.
  - `ROW` and `RANGE`:
    - Further limits the rows within the partition by specifying start and end points within the partition.
    - This is done by specifying a range of rows with respect to the current row either by logical association or physical association.

SQL

- `OVER()`:
  - `PARTITION BY`:
    - Divides the query result set into partitions. The window function is applied to each partition separately and computation restarts for each partition.
  - `ORDER BY`:
    - Defines the logical order of the rows within each partition of the result set, i.e., it specifies the logical order in which the window function calculation is performed.
  - `ROW` and `RANGE`:
    - Further limits the rows within the partition by specifying start and end points within the partition.
    - This is done by specifying a range of rows with respect to the current row either by logical association or physical association.
    - The `ROWS` clause limits the rows within a partition by specifying a fixed number of rows preceding or following the current row.

SQL

- `OVER()`:
  - `PARTITION BY`:
    - Divides the query result set into partitions. The window function is applied to each partition separately and computation restarts for each partition.
  - `ORDER BY`:
    - Defines the logical order of the rows within each partition of the result set, i.e., it specifies the logical order in which the window function calculation is performed.
  - `ROW` and `RANGE`:
    - Further limits the rows within the partition by specifying start and end points within the partition.
    - This is done by specifying a range of rows with respect to the current row either by logical association or physical association.
    - The `ROWS` clause limits the rows within a partition by specifying a fixed number of rows preceding or following the current row.
    - The `RANGE` clause logically limits the rows within a partition by specifying a range of values with respect to the value in the current row.

SQL

- `OVER()`:
  - `PARTITION BY`:
    - Divides the query result set into partitions. The window function is applied to each partition separately and computation restarts for each partition.
  - `ORDER BY`:
    - Defines the logical order of the rows within each partition of the result set, i.e., it specifies the logical order in which the window function calculation is performed.
  - `ROW` and `RANGE`:
    - Further limits the rows within the partition by specifying start and end points within the partition.
    - This is done by specifying a range of rows with respect to the current row either by logical association or physical association.
    - The `ROWS` clause limits the rows within a partition by specifying a fixed number of rows preceding or following the current row.
    - The `RANGE` clause logically limits the rows within a partition by specifying a range of values with respect to the value in the current row.
    - Preceding and following rows are defined based on the ordering in the `ORDER BY` clause.

SQL

- **Example**

SQL

- **Example**
  - Student grades with the average:

```
SELECT Student, Instructor, Lecture, Academic_year,
grade, AVG (grade) OVER (PARTITION BY Student)
FROM Grades;
```

OLAP Queries in MDX

- MDX $\longrightarrow$ Multidimensional expressions.

OLAP Queries in MDX

- MDX $\longrightarrow$ Multidimensional expressions.
- For OLAP queries, MDX is an alternative to SQL:

OLAP Queries in MDX

- MDX $\longrightarrow$ Multidimensional expressions.
- For OLAP queries, MDX is an alternative to SQL:

Academic_year	Instructor	AVG(Grade)
2011/2	Stefanowski	4.2
2011/2	Słowiński	4.1
2012/3	Stefanowski	4.0
2012/3	Słowiński	3.8
2013/4	Stefanowski	3.9
2013/4	Słowiński	3.6
2013/4	Dembczyński	4.8

OLAP Queries in MDX

- MDX $\longrightarrow$ Multidimensional expressions.
- For OLAP queries, MDX is an alternative to SQL:

Academic_year	Instructor	AVG(Grade)
2011/2	Stefanowski	4.2
2011/2	Słowiński	4.1
2012/3	Stefanowski	4.0
2012/3	Słowiński	3.8
2013/4	Stefanowski	3.9
2013/4	Słowiński	3.6
2013/4	Dembczyński	4.8

$\downarrow$

AVG(Grade)	Academic_year		
Name	2011/2	2012/3	2013/4
Stefanowski	4.2	4.0	3.9
Słowiński	4.1	3.8	3.6
Dembczyński			4.8

MDX

- MDX query:

```
SELECT {[Academic Year].[2011/2],[Academic
Year].[2012/13],[Academic Year].[2013/14]} ON COLUMNS,
{[Instructor].[Stefanowski],[Instructor].[Slowinski],
[Instructor].[Dembczynski]} ON ROW
FROM PUT
WHERE ([Measures].[Average Grades])
```

- Seems to be similar to **SQL**, but in fact it is quite **different**!

Outline

Summary

- ETL process is a strategic element of data warehousing.
- Main concepts: extraction, transformation and integration, load, data warehouse refreshment and metadata.
- New emerging technology ...
- OLAP systems: ROLAP, MOLAP and HOLAP.
- Two main approaches for querying data warehouses.
  - ROLAP servers: SQL and its OLAP extensions.
  - MOLAP servers: MDX.

Bibliography

- J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, second edition edition, 2006
- Mark Whitehorn, Robert Zare, and Mosha Pasumansky. *Fast Track to MDX*. Springer, 2002