DATABASES IN BIOLOGY AND MEDICINE

Relational Databases

Biomedical Engineering

Adam Piórkowski

pioro@agh.edu.pl

http://home.agh.edu.pl/~pioro/dyd/

http://home.agh.edu.pl/~pioro/dyd/DBiBaM/

© AP, all rights reserved

Kraków, 2025–2026

Introduction

A database is a collection of data managed by a database management system.

A database management system (DBMS) is software that provides the following services:

- - data organization and classification;
- - persistent storage of large (theoretically unlimited) amounts of data;
- - data browsing, insertion, deletion, and modification;
- - protection against unauthorized data access and modification;
- - control of concurrent access to resources;
- - protection against erroneous operations;
- - data recovery after failures or emergency situations.

Introduction

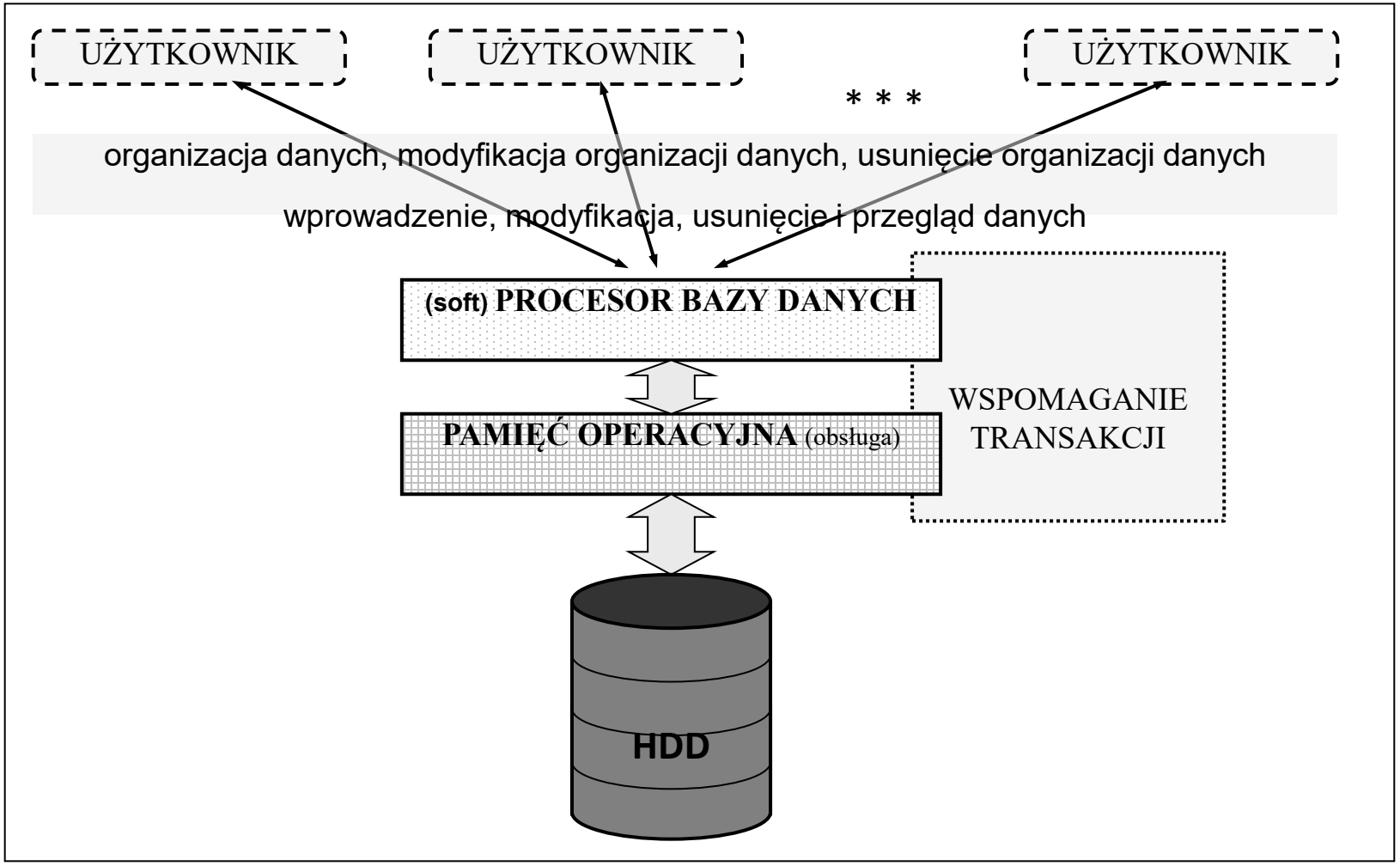
A data bank is a database used for data storage.

A data repository is a database of archived data.

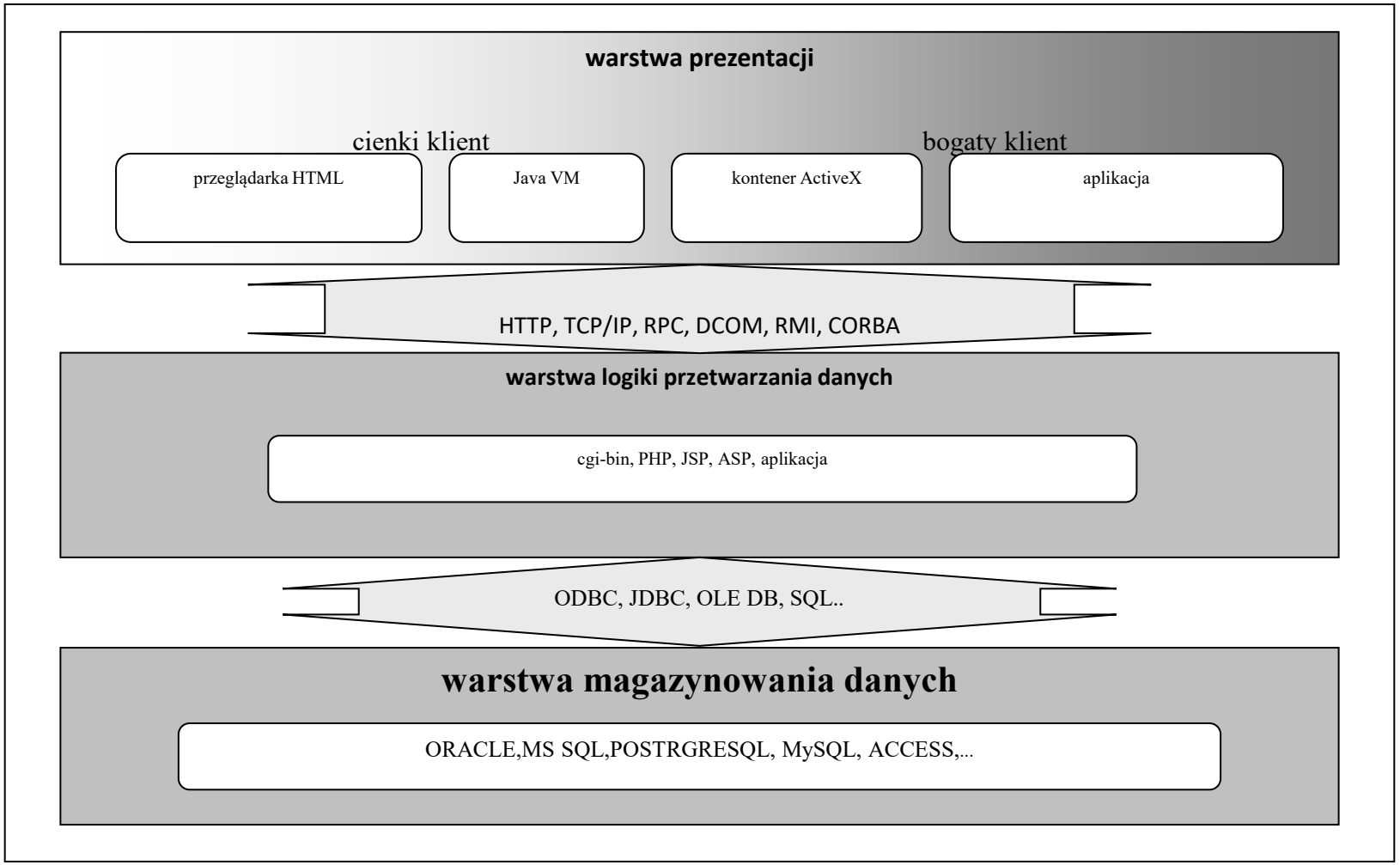
A data warehouse is a higher-level database that periodically retrieves data from subordinate operational databases and contains verified, optionally aggregated and selected data. It is intended for rapid subject-oriented retrieval and is often not a relational database.

Data mining is the process of analysing data in order to identify relationships that are not readily apparent to humans.

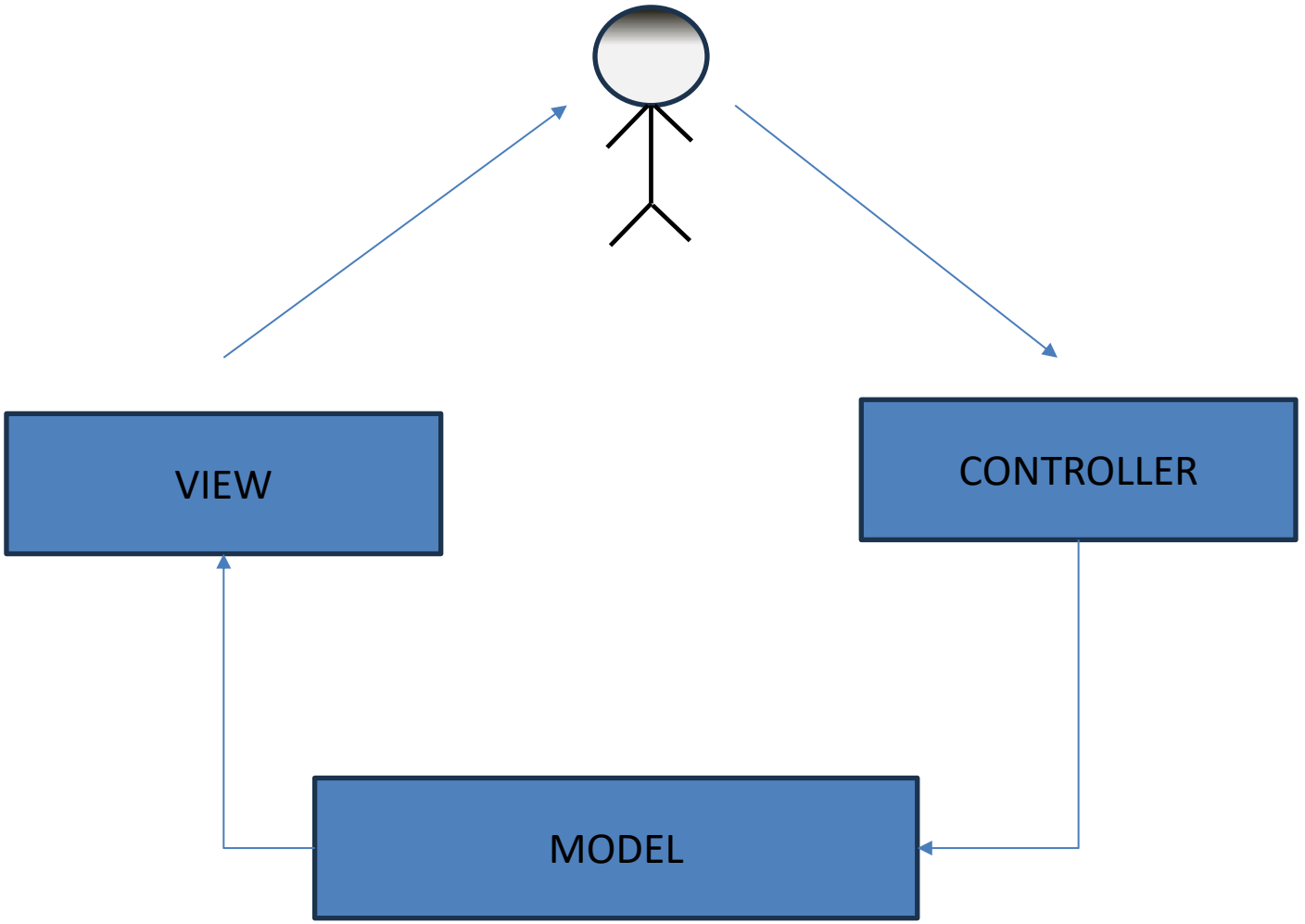
Components of a Database Management System



Three-Tier Architecture



Model–View–Controller



Types of Databases – General Approach

- **Simple databases – data are stored in tables that are not linked by relationships. This is usually a dedicated solution.**
- **Analytical databases – collect data for browsing and analysis.**
- **Operational databases – collect data and allow them to be modified at any time.**
- **Passive databases – databases in which actions are triggered by the user (HADP – Human Active Database Passive).**
- **Active databases – databases that can trigger actions themselves (DAHP – Database Active Human Passive).**

Types of Databases – Data Organization

- **Hierarchical databases – a database concept with a tree structure in which each node contains a record (data set) and references to a finite number of other records related according to a hierarchy (parent–child). Links between records are unidirectional and have logical meaning.**

  - Advantages:
  - - simple structure;
- - fast data access.

  - Disadvantages:
- - limited ability to implement relationships (1:1, 1:N);
- - difficult database modification.

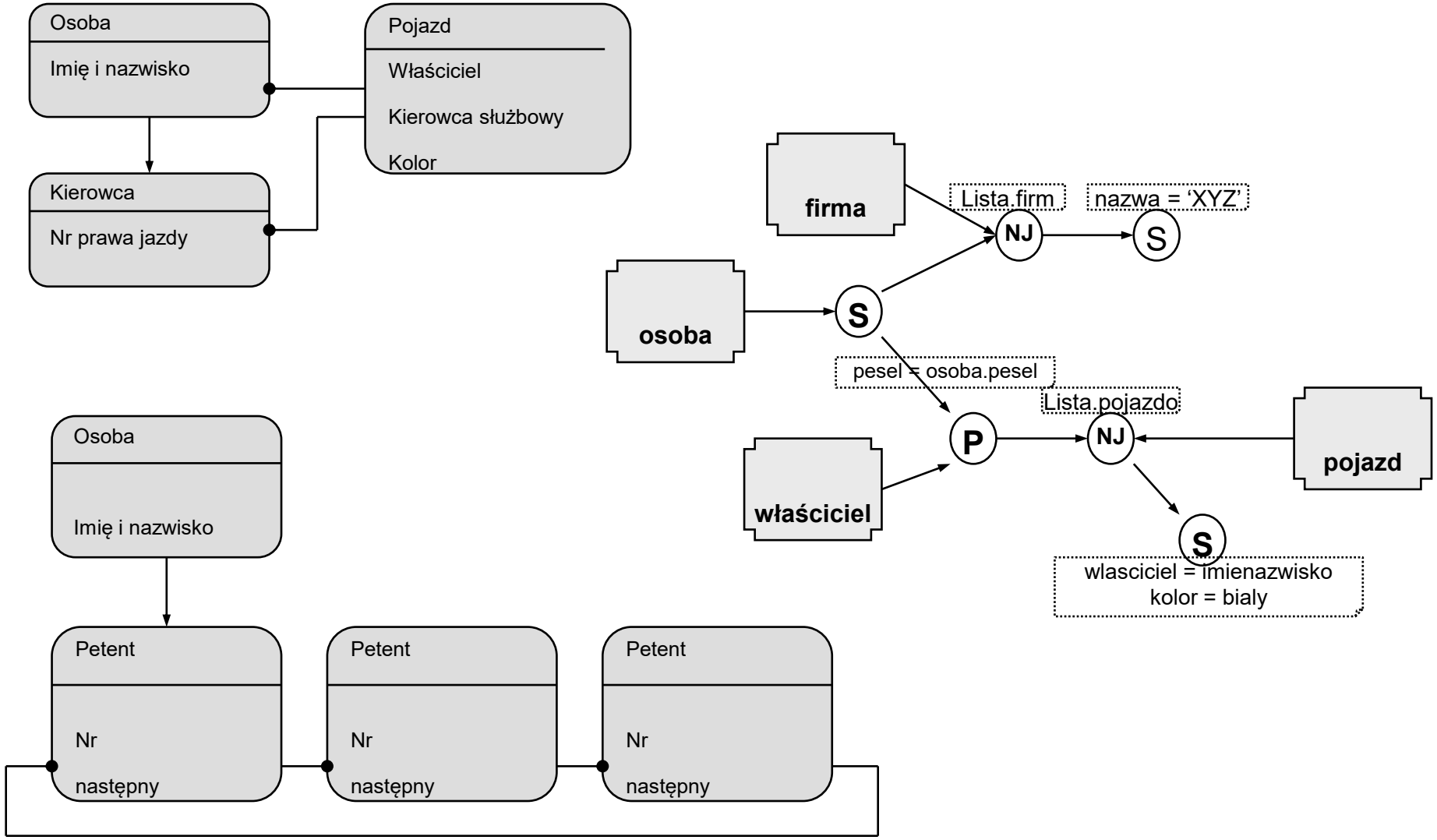
Types of Databases – Data Organization

- **Relational databases – currently the most widely used type of database, proposed by E. F. Codd in 1970. The relational database model stores data in tables called relations. Each relation consists of a set of attributes. Table entries (tuples) are stored in rows, with at most one value for a given attribute in each row. The following relationships between relations are possible: 1:1, 1:N, N:1, M:N.**

Types of Databases – Data Organization

- **Object-oriented databases – this type of database is not strictly defined. It is based on object-oriented programming paradigms: data are classified as objects of particular classes, with appropriate mechanisms of encapsulation and inheritance. An advantage is the ability to retrieve data easily and quickly; a disadvantage is dependence on the specific platform on which the database is developed (an attempt at standardization is the ODL language).**

Object-Oriented Databases



Types of Databases – Data Organization

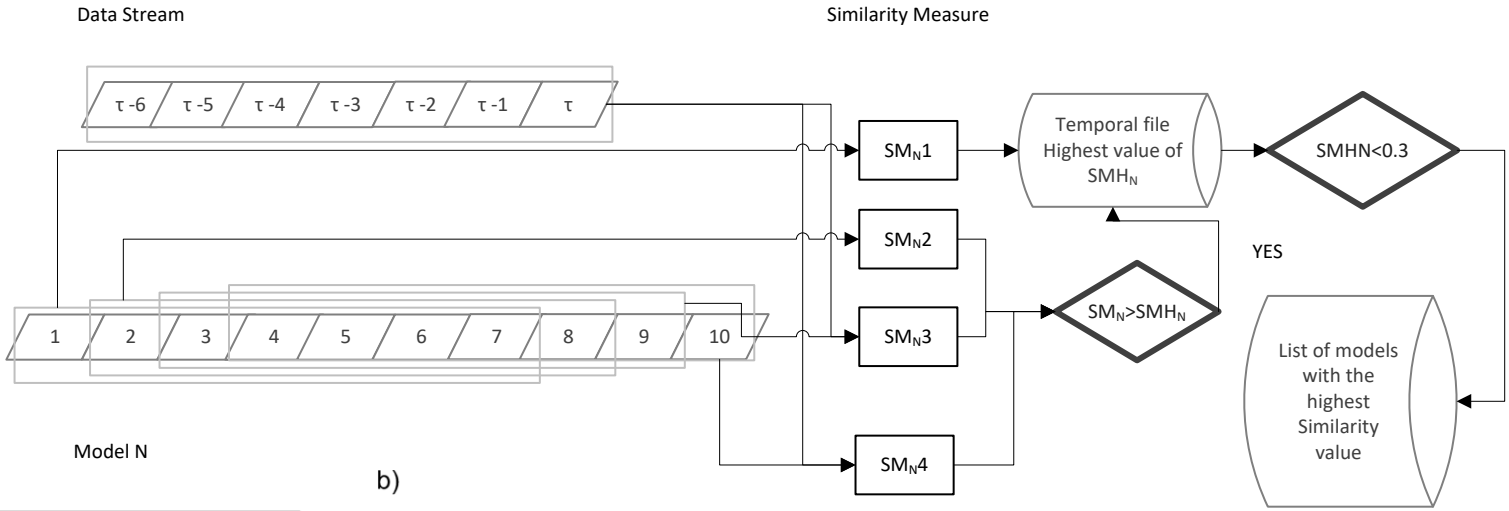
- **Object-relational databases – a hybrid of object-oriented and relational databases, allowing non-standard data (multimedia, abstract data, etc.) to be stored while retaining the ease of use of a relational database.**
- **Temporal databases – databases in which data are additionally associated with temporal attributes. Data may be valid at particular points in time, over time intervals, or over sets of points and intervals.**

Types of Databases – Data Organization

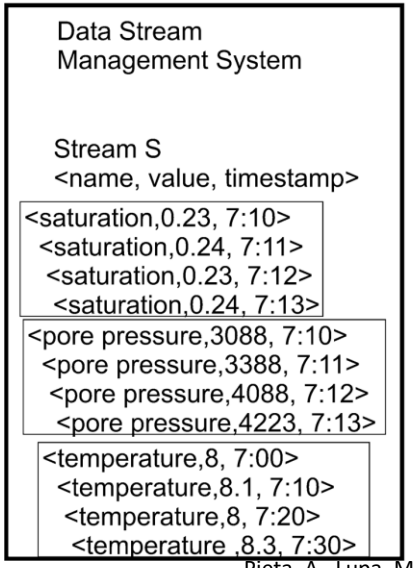
- **Stream databases – active databases oriented towards data represented as streams.**

  - Characteristic features of stream databases:
  - - the user does not supervise data acquisition;
  - - data may arrive continuously; processing focuses mainly on the most recent data, which may be archived or discarded;
  - - data streams have no theoretical size limit;
  - - a built-in data acquisition module;

- - a stream query language (declarative, object-oriented, or procedural) that supports operations on data streams; for example, a query may

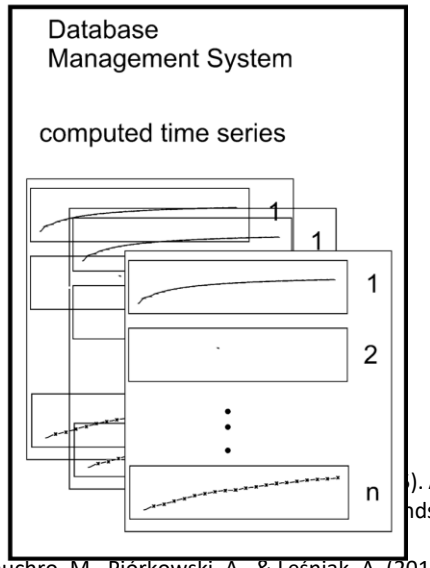


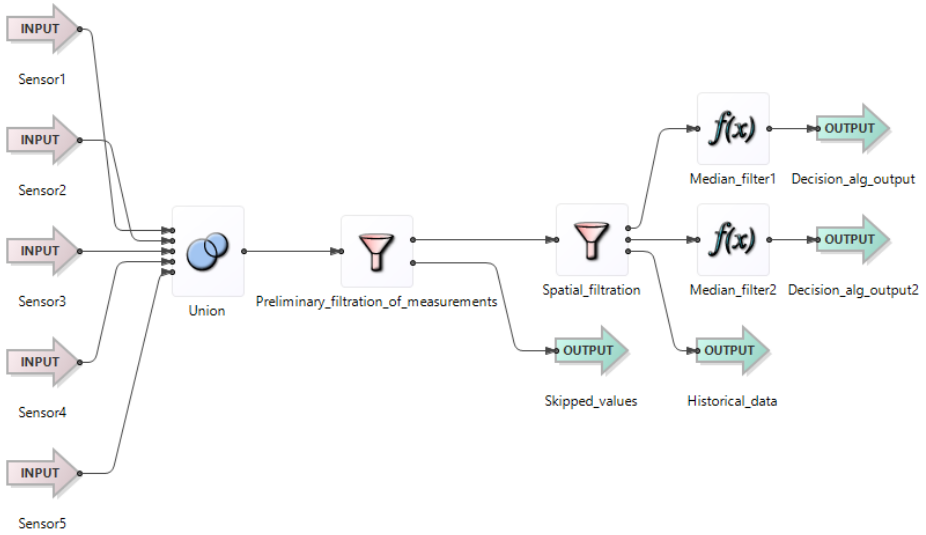
a)



Stream Databases

b)





). A concep
nds in Datal

Pięta, A., Lupa, M., Chuchro, M., Piórkowski, A., & Leśniak, A. (2014). A Model of a System for Stream Data Storage and Analysis Dedicated to Sensor Networks of Embankment Monitoring. In Computer Information Systems and Industrial Management (pp. 514-525).

Types of Databases – Data Organization

- **XML Databases – ….**

  These are either standalone databases that store data in XML format or extensions of, for example, relational databases, allowing data stored in a column as text/binary data to be processed and interpreted by extensions of the query language.

- XML (eXtensible Markup Language) is a universal, textual, standardized, hardware-independent format for organizing and storing data. The basic element used to store and organize data is the so-called tag. Structure of an XML document:
  - - a header specifying the XML version and encoding;
  - - a root tag enclosing the entire document.

- XML Schema Definition (XSD) is a method of attaching information about data structure to a document.
- XPath is a language used for navigation and operations on XML data.
- XQuery is a standardized language for operating on XML documents that incorporates (extends) XPath.

XML Databases

```
<?xml version="1.1" encoding="ISO-8859-2"?>
<tag_glowny>
<nametagu>dane</nametagu>
<tagpustybezdanychiparametrow/>
<tagzparametrami parametr=”pierwszy” opcja=”druga”>dane</tagzparametrami>
<danebinarne dt:dt="binary.base64">fs54Dh6745GgU764s</danebinarne>
<danebinarnenasze>fs54Dh6745GgU764s</danebinarnenasze>
</tag_glowny>
```

```
<?xml version="1.0" encoding="ISO-8859-2" standalone="yes"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:simpleType name="custom-date-type">
                <xs:restriction base="xs:string">
                        <xs:pattern value="[0-9]{2}/[0-9]{2}/[0-9]{4}"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:element name="id_firmy" type="xs:int">
                <xs:annotation>
                        <xs:documentation>number firmy</xs:documentation>
                </xs:annotation>
        </xs:element>
        <xs:element name="wojewodztwo">
                <xs:annotation>
                        <xs:documentation>informacje skategoryzowane</xs:documentation>
                </xs:annotation>
                <xs:simpleType>
                        <xs:restriction base="xs:string">
                                <xs:enumberation value="malopolskie"/>
                                <xs:enumberation value="podkarpackie"/>
```

JSON Format (JavaScript Object Notation)


```
{
  "Description": "Basic data",
  "Display enabled": 1,
  "Block size": [512,512,288],
  "LineX": [158, 158, 158, 158, 158, 155, 155, 155, 155],
  "LineY": [417, 417, 417, 417, 417, 417, 417, 417, 417],
  "Projections": [ {"name": "4 chambers","a": -40,"b": 0,"c": 5},
  {"name": "2 chambers","a": -55,"b": 0,"c": 90},
  {"name": "1 chamber","a": -60,"b": 0,"c": 70} ],
  "status": "2",
  "Errors": ""
}
```

Types of Databases – Data Organization

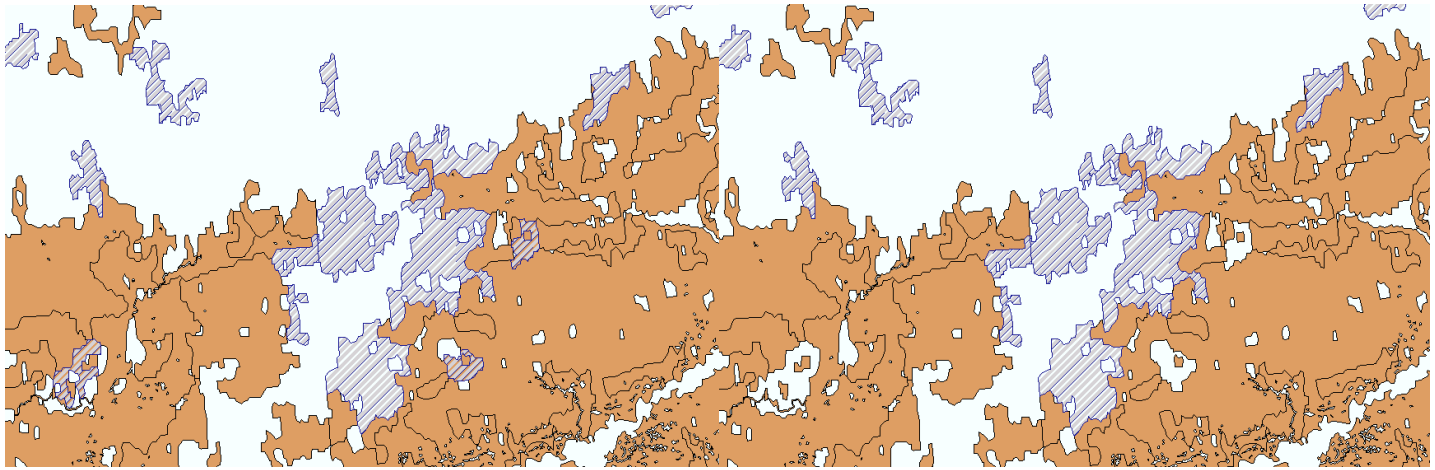
- **‘Spatial databases’ – databases equipped with a dedicated system for storing and processing spatial data.**
- Primary application – storage of spatial data.
- Extension of SQL for processing such data.

```
INSERT INTO Gminy (id_gminy, Nazwa_gminy, obszar)
 VALUES (1, 'Staropolska', GeomFromText('POLYGON((6 3, 6 19,
 21 19, 21 3, 6 3))'));

INSERT INTO Drogi(id_drogi, przebieg)
 VALUES (1, GeomFromText('LINESTRING(0 10, 15 10, 25 13)', -1));
```

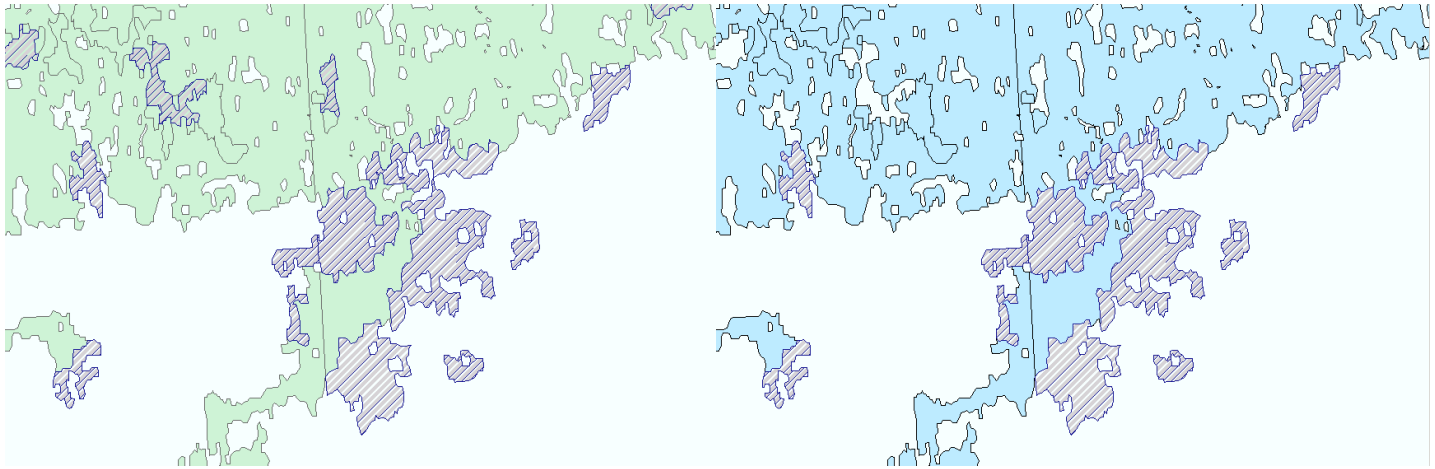
```
mysql> SELECT Nazwa_gminy, Area(obszar) FROM Gminy;
+-------------+--------------+
| Nazwa_gminy | Area(obszar) |
+-------------+--------------+
| Staropolska |          240 |
+-------------+--------------+
1 row in set (0.00 sec)
```

```
psql=# SELECT Nazwa_gminy, Area(obszar) FROM Gminy;
 nazwa_gminy | area
-------------+------
 Staropolska |  240
(1 row)
```

Spatial Databases



Input layers: trees and swamp (L), disjoint of input layerers (R)

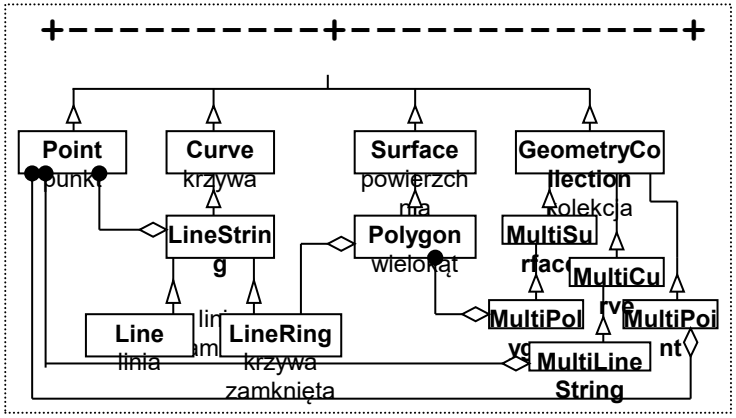


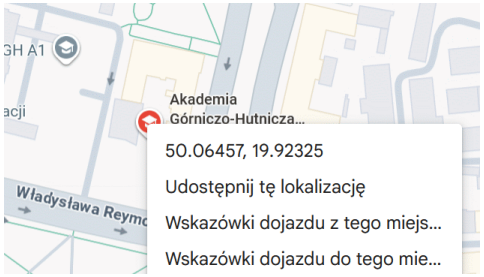
Input layers: tundra and swamp (L), disjoint of input layerers (R)

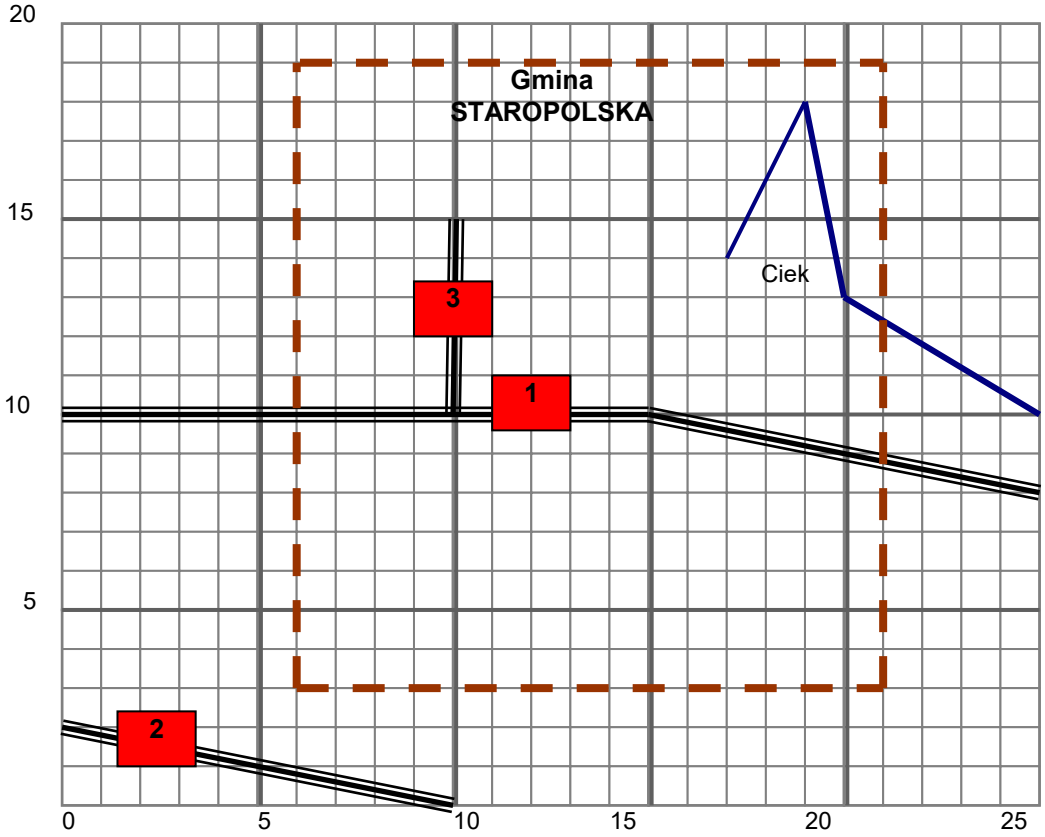
Lupa, M., & Piórkowski, A. (2014). Spatial query optimization based on transformation of constraints. In Man-Machine Interactions 3 (pp. 621-629).

Spatial Databases

```
-- common part
SELECT id_drogi, Nazwa_gminy FROM Drogi, Gminy
                  WHERE Intersects(obszar, przebieg);
+----------+-------------+
| id_drogi | Nazwa_gminy |
+----------+-------------+
|        1 | Staropolska |
|        3 | Staropolska |
+----------+-------------+
```







Piórkowski, A. (2011). Mysql spatial and postgis-implementations of spatial data standards. Electronic Journal of Polish Agricultural Universities, 14(1), 1-8.

Database Management Systems

- **SQL databases – relational databases**
  - Free: MySQL, PostgreSQL, SQLite
  - Commercial: Oracle, MS SQL Server, IBM DB2
- **Databases for less advanced office applications**
  - MS Access – commercial, also includes SQL
  - OpenOffice Base / LibreOffice Base – free alternatives
- **NoSQL – databases intended for projects involving enormous amounts of data (Big Data), focused on performance and often implemented according to the <key, value> model**

NoSQL Databases

- A newer type of database that does not use the relational schema as its primary model.
- Consequently, the SQL language standard is not used.
- Each system has its own language.
- Different data models, e.g.:
  - - key–value databases;
  - - document databases (e.g. JSON-based),

Data Type Unification

- **Data Types**
  - numeric data (integers, decimals, floating-point numbers);
  - operations on numeric data – conversion: casting, rounding, truncation; arithmetic operations
  - character data, Unicode
  - time measures (date, time)
  - NULL
  - BLOB
  - multimedia data (images, sounds, video, streams)
  - data in special formats, e.g. XML

Multi-user Access

- **Data Access (Users)**
- Database systems provide different levels of data access. The most common user roles in relational databases are:
  - administrator – typically has privileges to create, connect to, and delete databases, create and delete tables, and perform arbitrary operations on data;
  - user – typically has read-only access to the database, optional privileges to create and delete tables (e.g. granted by an administrator), and full or restricted rights to manipulate table data.

- Concurrency in Databases

Database systems support concurrent access to data. Individual operations are atomic. Sequences of operations can be protected by transaction mechanisms with different isolation levels. Locks are available.

Selected Database System Functions

- **Reports – used to present the contents of a database, a fragment of it, or the results of a database query in an organized form. They do not modify data.**
- **Forms – used to enter or modify data in a database through interaction with the user. They help ensure the correctness of the supplied data.**
- **Queries – implement a user's request to the database and may be executed interactively. Queries allow data to be displayed, analysed, searched, and modified. Their result is a set of records satisfying the query expression.**

  - Modification queries:
  - - delete queries – delete a set of tuples from a table (or tables);
  - - append queries – add a set of tuples to a table (or tables);
  - - update queries – perform operations on selected attributes of all tuples in selected tables;
- - make-table queries – e.g. create a backup copy of a table.

- Crosstab queries – operate on at least two tables; their result is the Cartesian product of the table data.

Selected Database System Functions

- **Triggers – allow a user (or application) to be notified and/or an action to be performed when an event occurs in the database.**
- Stored procedures – procedures stored on and executed by the server, written in SQL or in the server's internal language (in which case they may be precompiled into bytecode).
- Archiving (backups) – makes it possible to create an image of the current state of data for recovery or migration.

RELATIONAL DATABASES

E. F. Codd, 1970

-relational data structures

-relational algebra operators

-data integrity constraints

- **Formal Representation of the Relational Model**
- ***domain = attribute = a set of elements of a given type (integers, floating-point numbers, text, etc.)***

Let us assume the set of attributes { D1, D2, ..., DN }.

- Cartesian product of attributes: D1 × D2 × ... × DN
- A relation is any subset of the Cartesian product formed from the list of domains.

RELATIONAL DATABASES

Cartesian Product – Example:

D1 = {a, b}, D2 = {c, d}
D1 × D2 = { (a,c), (b,c), (a,d), (b,d) }

Relations – Example:

D1 = {a, b}, D2 = {c, d}
R1 = { (a,c), (b,c), (b,d) }
R2 = { (a,c) }
R3 = { (a,d), (b,c) }
R4 = { (a,c), (b,c), (a,d), (b,d) }
…

RELATIONAL DATABASES

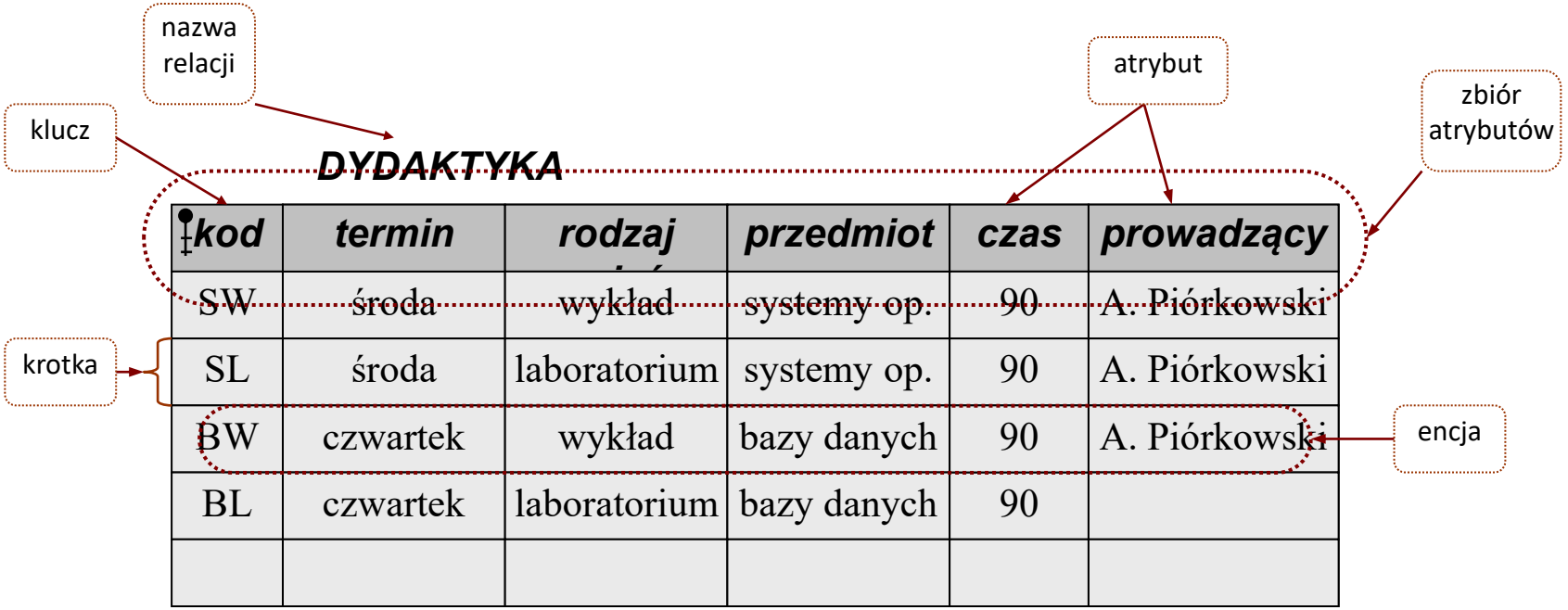
Cartesian Product – Example:

Dfirst_name = {Adam, Ewa}, Dlast_name = {Nowak, Kowal}
Dfirst_name × Dlast_name = { (Adam, Nowak), (Adam, Kowal),
(Ewa, Nowak), (Ewa, Kowal) }

Relations – Example:

Rwomen = {(Ewa, Nowak), (Ewa, Kowal) }
RNowakFamily = {(Ewa, Nowak), (Adam, Nowak), }

Relational Database Concept



kod	termin	rodzaj	przedmiot	czas	prowadzący
SW	środa	wykład	systemy op.	90	A. Piórkowski
SL	środa	laboratorium	systemy op.	90	A. Piórkowski
BW	czwartek	wykład	bazy danych	90	A. Piórkowski
BL	czwartek	laboratorium	bazy danych	90	

Relational Database Concept

An entity is an elementary unit of data describing an object with certain properties.

An attribute is a characteristic shared by entities belonging to a given entity set.
Each relation has a set of attributes; each attribute corresponds to a column in the table of that relation.

Relation schema = relation name + set of attributes.
Each row of a table corresponds to exactly one record.

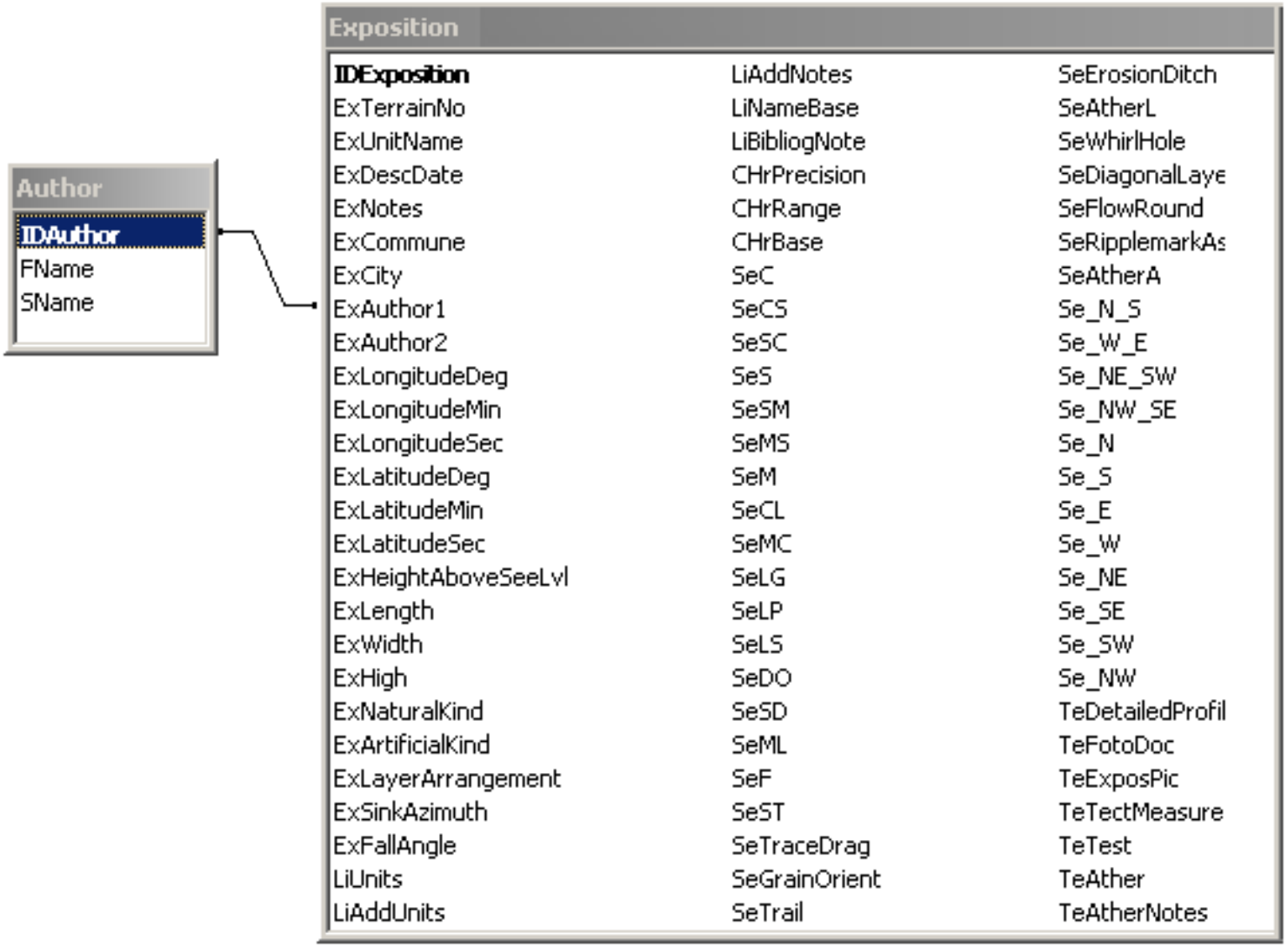
relation schema

DYDAKTYKA
kod
termin
rodzaj zajęć
przedmiot
czas
prowadzący

A tuple is a table row containing data and corresponding to one record.
Relation data consist of the set of all tuples.
Tuples do not necessarily have unique data (optional).
The order of tuples is arbitrary and is usually the order in which they were entered.

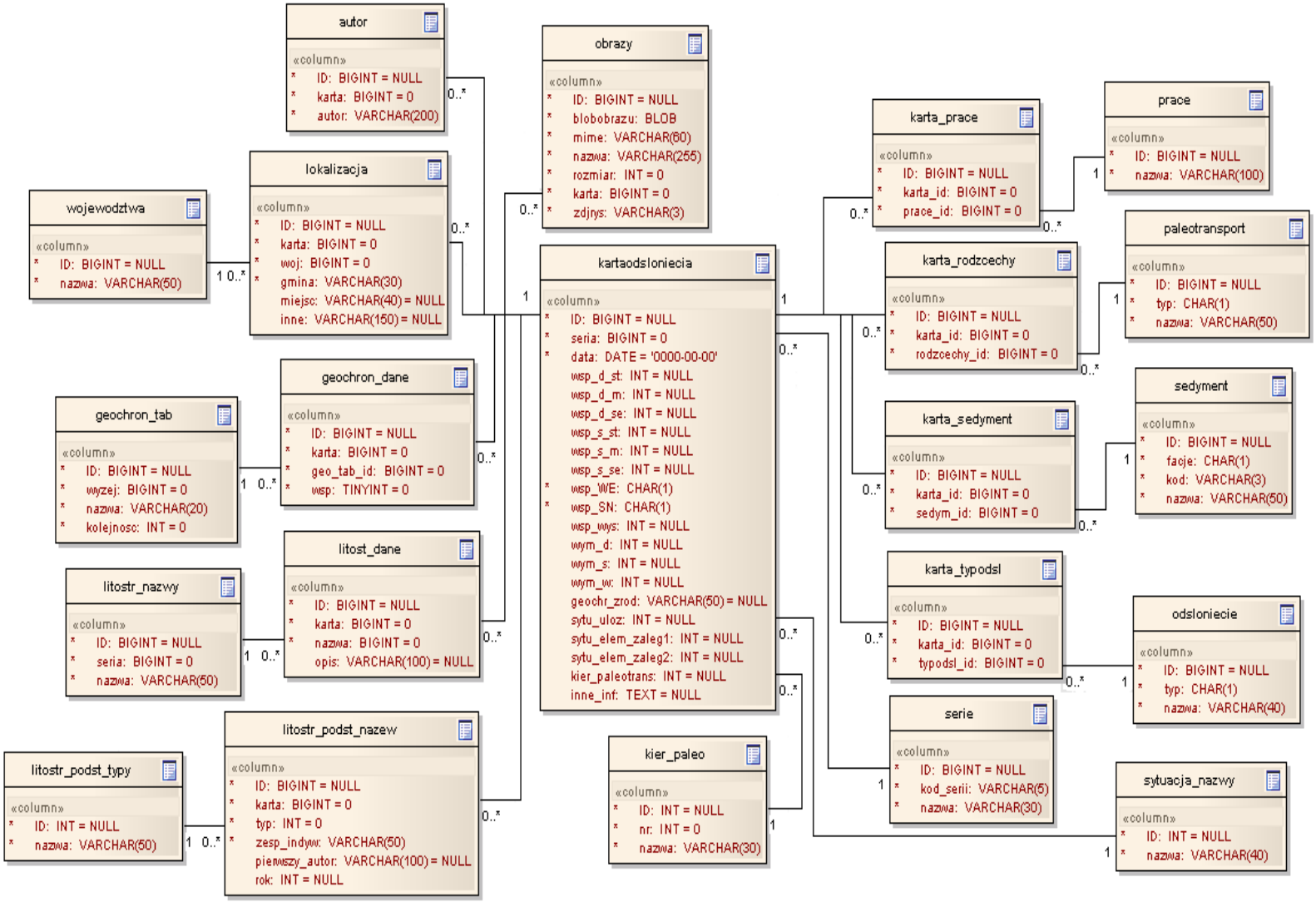
A key is an attribute (or set of attributes) whose values are unique for all tuples.

EXAMPLES



Kotlarczyk J., Krawczyk A., Leśniak T., Słomka T. : GeoKarpaty Geological Database for the Polish Flysch Carpathians. AGH WGGiOŚ Publishing, Kraków 1997
Gajda G., Piórkowski A.: Possibilities for Constructing Geological Data Warehouses. Databases – Development of Methods and Technologies – Security, Selected Technologies and Applications. WKŁ, Warsaw 2008.

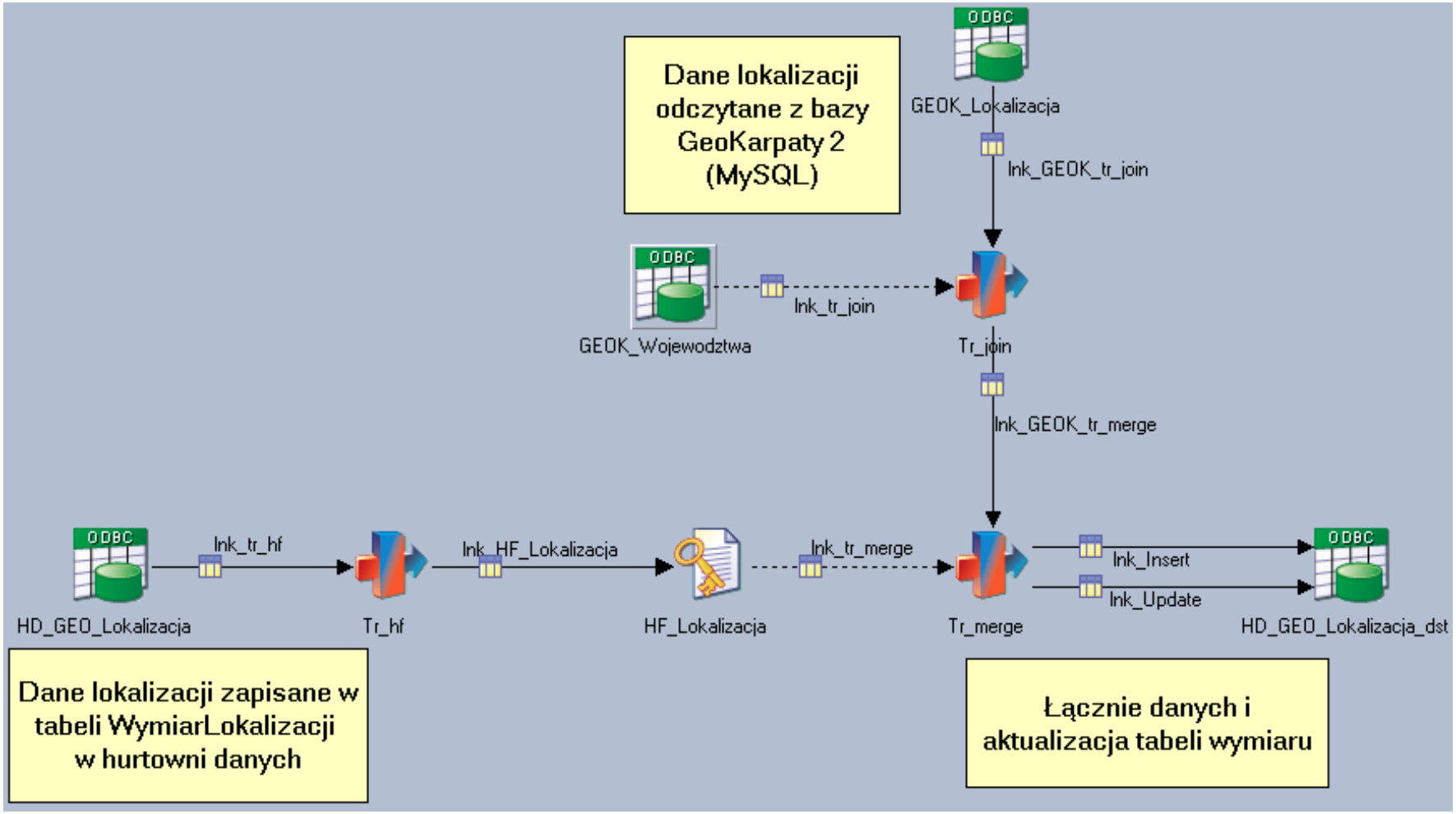
EXAMPLES



Malec O., Kyc M., Piórkowski A.: GeoKarpaty II Web Database of Geological Exposures. Proceedings of the First Polish Geological Congress, Kraków 26–28 June 2008. Polish Geological Society, 2008.

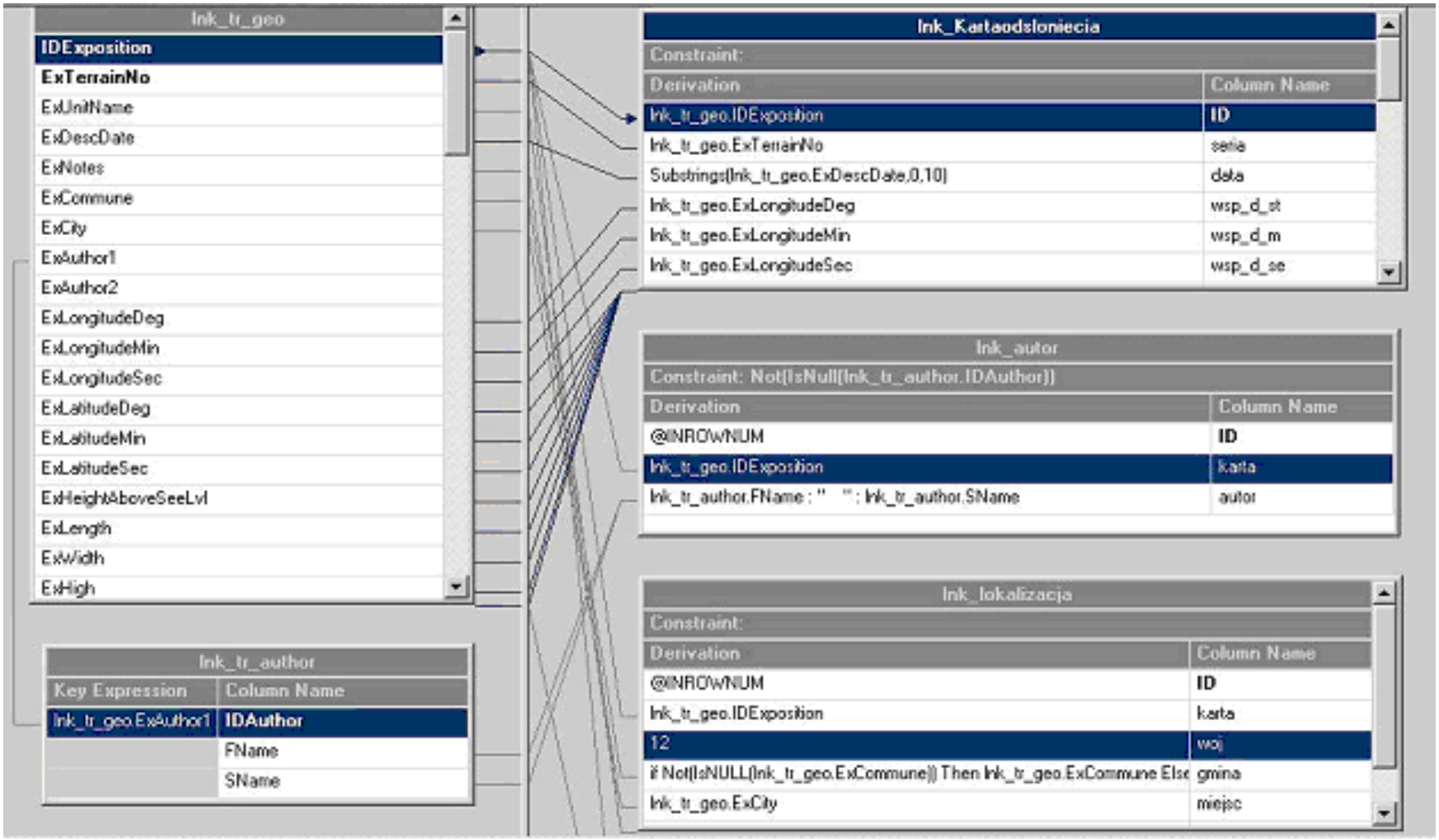
Gajda G., Piórkowski A.: Possibilities for Constructing Geological Data Warehouses. Databases – Development of Methods and Technologies – Security, Selected Technologies and Applications. WKŁ, Warsaw 2008.

EXAMPLES



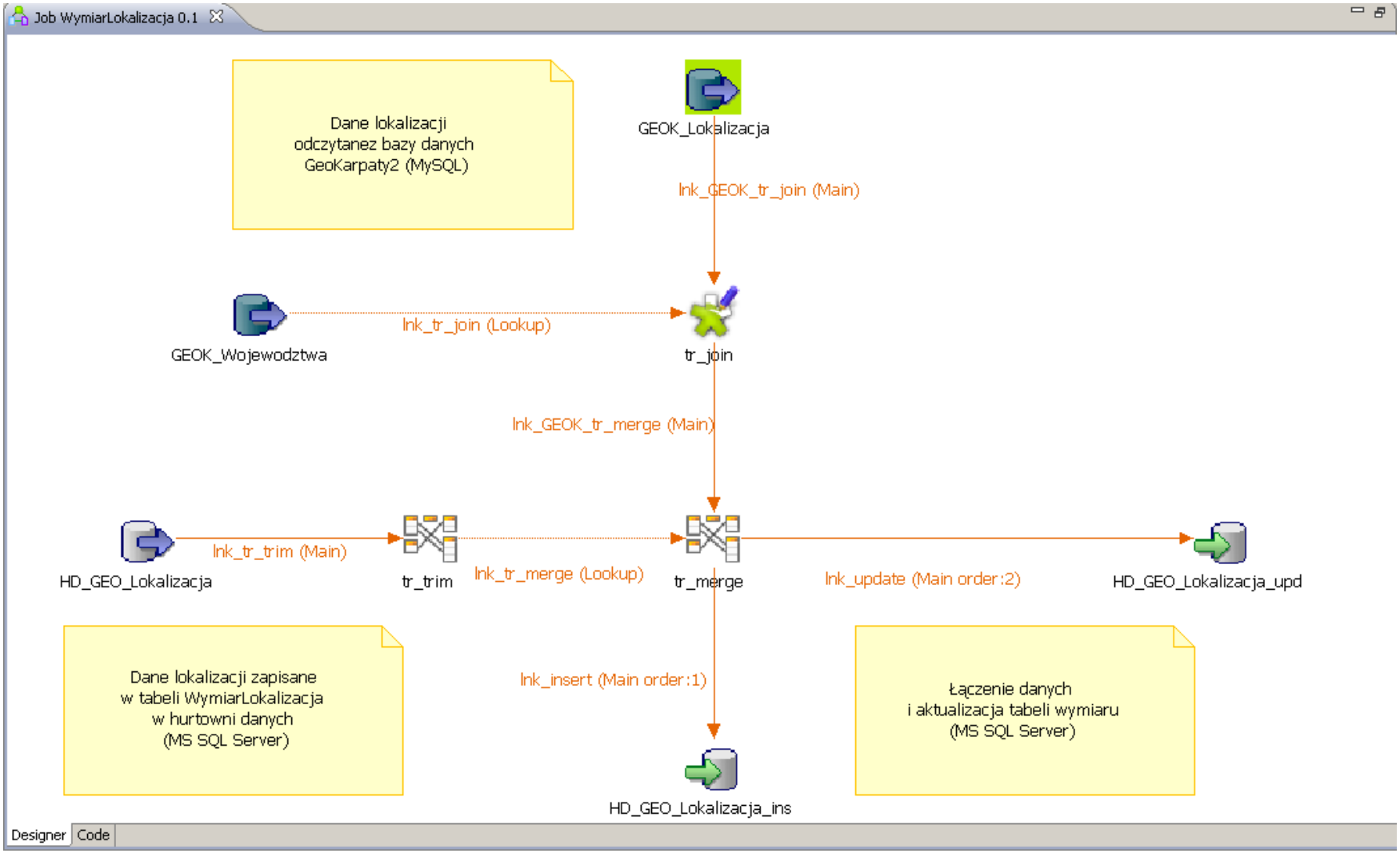
Piórkowski A., Gajda G.: Construction of a Multidimensional Geological Database. Studia Informatica 2009, Vol. 30 No 2B (84)

EXAMPLES



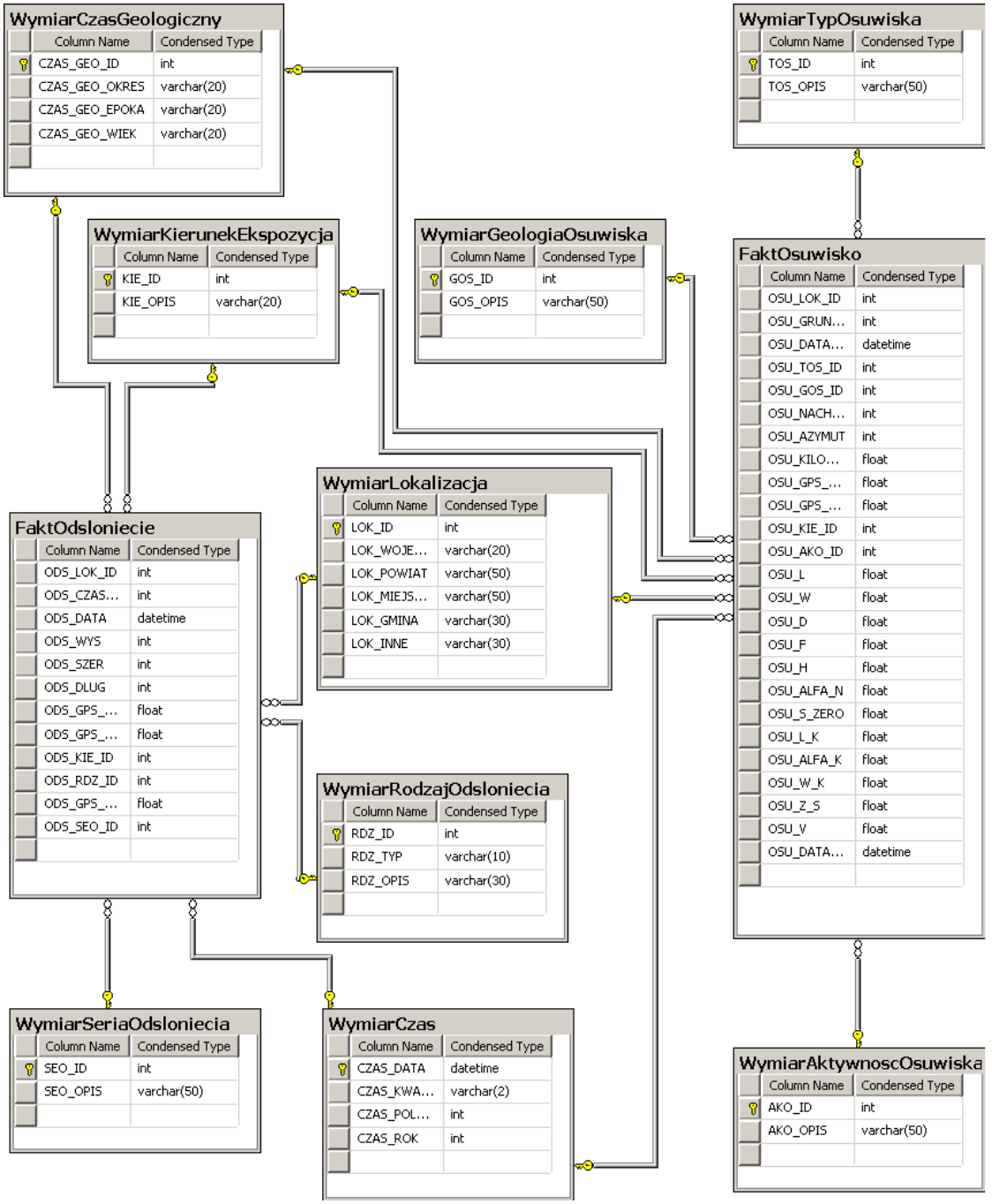
Piórkowski A., Gajda G.: Construction of a Multidimensional Geological Database. Studia Informatica 2009, Vol. 30 No 2B (84)

EXAMPLES



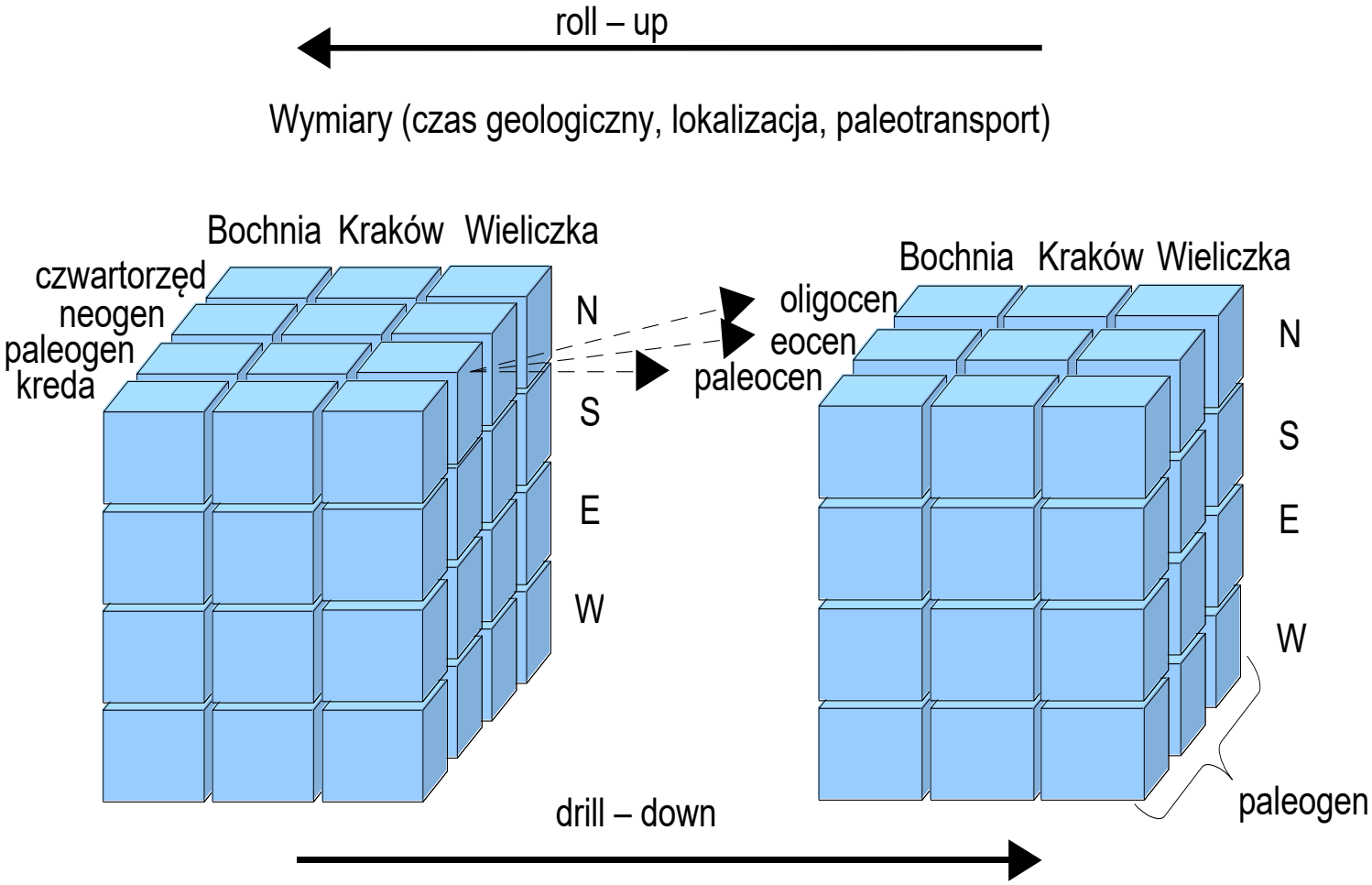
Piórkowski A., Gajda G.: Construction of a Multidimensional Geological Database. Studia Informatica 2009, Vol. 30 No 2B (84)

EXAMPLES



Piórkowski A., Gajda G.: Construction of a Multidimensional Geological Database. Studia Informatica 2009, Vol. 30 No 2B (84)

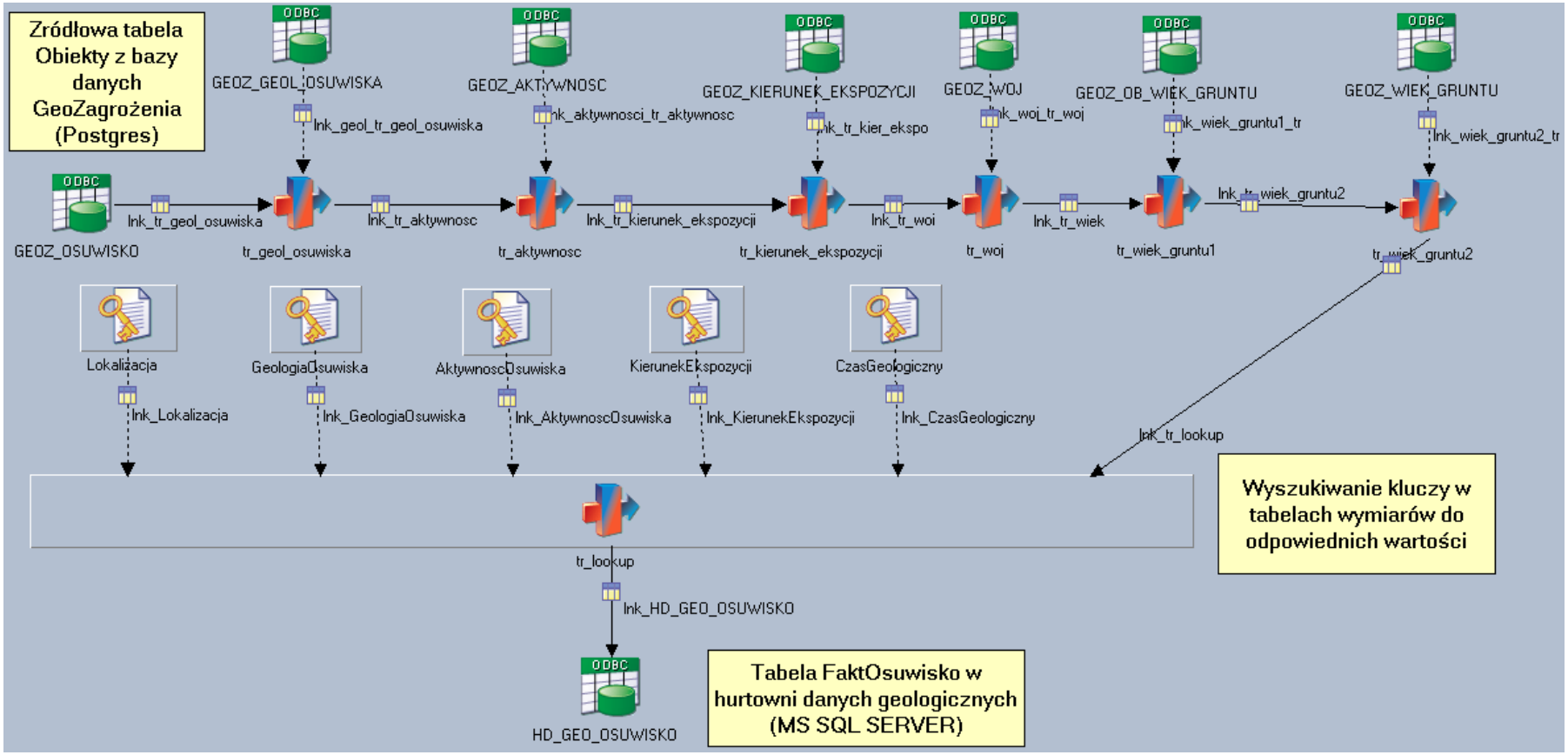
EXAMPLES



Hierarchia atrybutu WymiarCzasGeologiczny: Okres -> Epoka - >Wiek

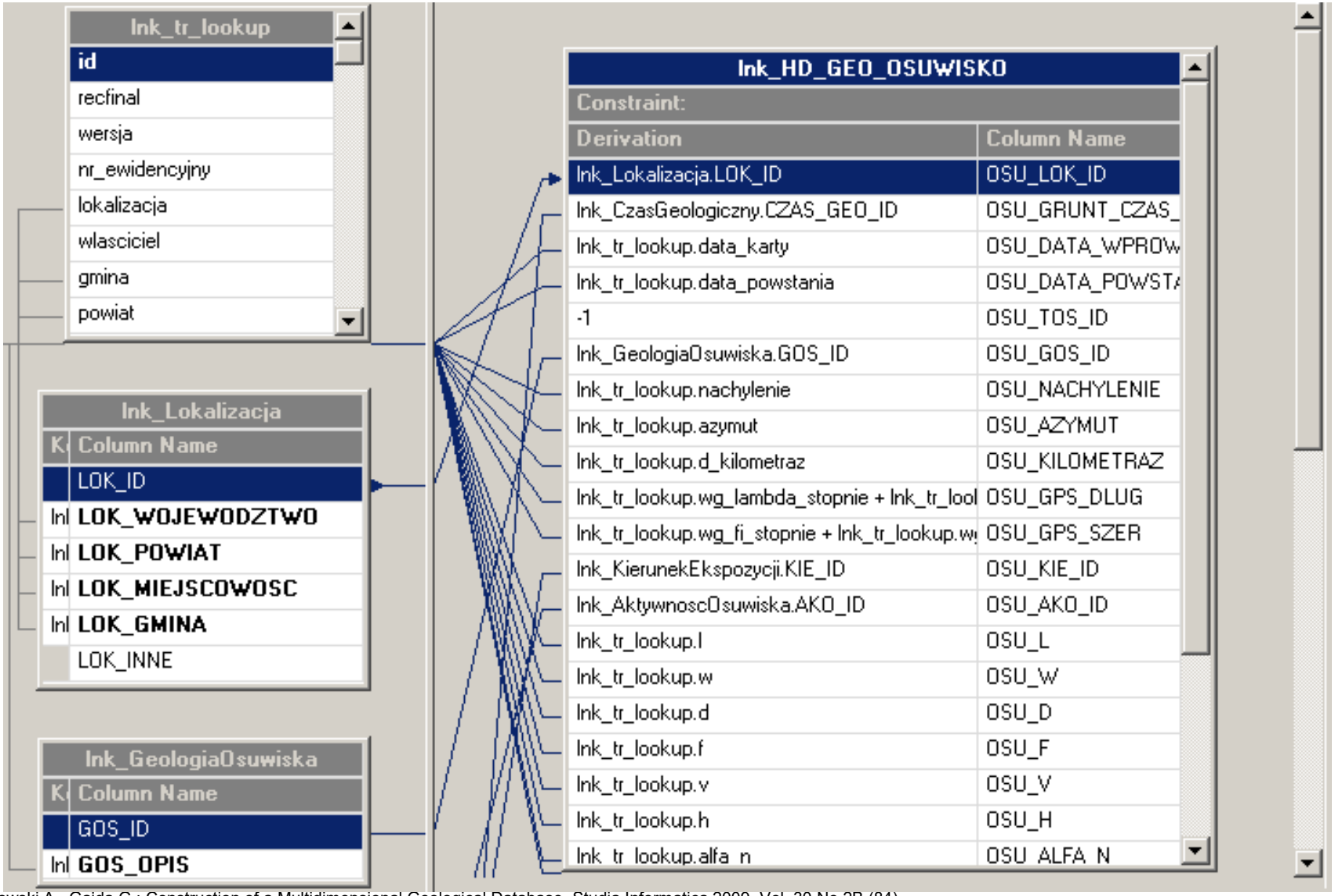
Piórkowski A., Gajda G.: Construction of a Multidimensional Geological Database. Studia Informatica 2009, Vol. 30 No 2B (84)

EXAMPLES

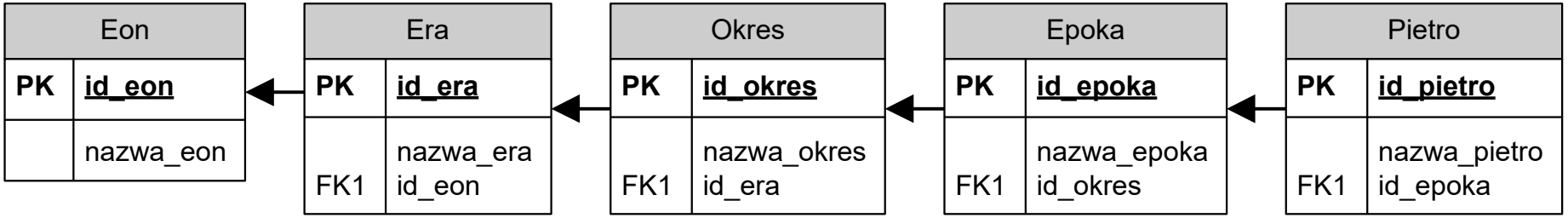


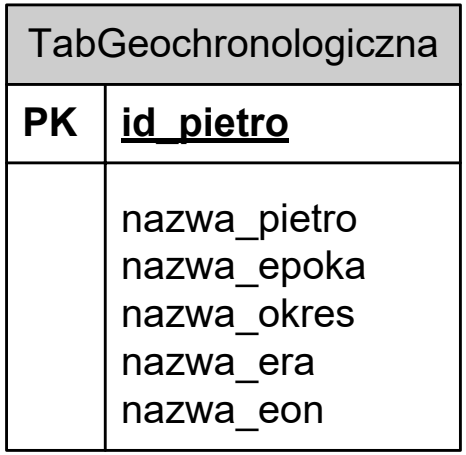
Piórkowski A., Gajda G.: Construction of a Multidimensional Geological Database. Studia Informatica 2009, Vol. 30 No 2B (84)

EXAMPLES


lnk_tr_lookup
id
recfinal
wersja
nr_ewidencyjny
lokalizacja
wlasciciel
gmina
powiat
lnk_Lokalizacja
K
Column Name
LOK_ID
lnl LOK_WOJEWODZTWO
lnl LOK_POWIAT
lnl LOK_MIEJSCOWOSC
lnl LOK_GMINA
LOK_INNE
lnk_GeologiaOsuwiska
K
Column Name
GOS_ID
lnl GOS_OPIS
lnk_HD_GEO_OSUWISKO
Constraint:
Derivation
Column Name
lnk_Lokalizacja.LOK_ID
OSU_LOK_ID
lnk_CzasGeologiczny.CZAS_GEO_ID
OSU_GRUNT_CZAS_
lnk_tr_lookup.data_karty
OSU_DATA_WPROW
lnk_tr_lookup.data_powstania
OSU_DATA_POWSTA
-1
OSU_TOS_ID
lnk_GeologiaOsuwiska.GOS_ID
OSU_GOS_ID
lnk_tr_lookup.nachylenie
OSU_NACHYLENIE
lnk_tr_lookup.azymut
OSU_AZYMUT
lnk_tr_lookup.d_kilometraz
OSU_KILOMETRAZ
lnk_tr_lookup.wg_lambda_stopnie + lnk_tr_lool
OSU_GPS_DLUG
lnk_tr_lookup.wg_fi_stopnie + lnk_tr_lookup.w
OSU_GPS_SZER
lnk_KierunekEkspozycji.KIE_ID
OSU_KIE_ID
lnk_AktywnoscOsuwiska.AKO_ID
OSU_AKO_ID
lnk_tr_lookup.l
OSU_L
lnk_tr_lookup.w
OSU_W
lnk_tr_lookup.d
OSU_D
lnk_tr_lookup.f
OSU_F
lnk_tr_lookup.v
OSU_V
lnk_tr_lookup.h
OSU_H
lnk_tr_lookup.alfa_n
OSU_ALFA_N

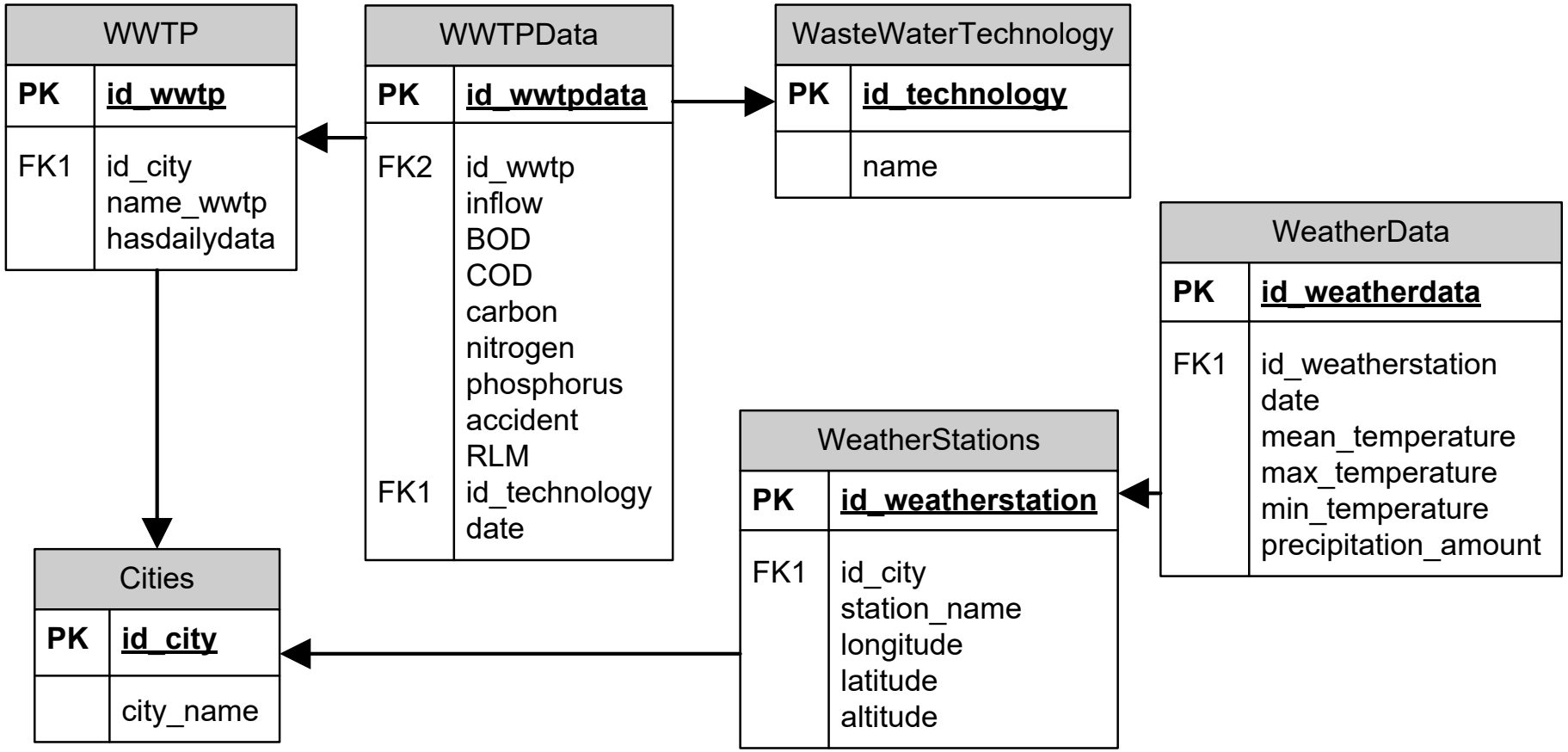
EXAMPLES



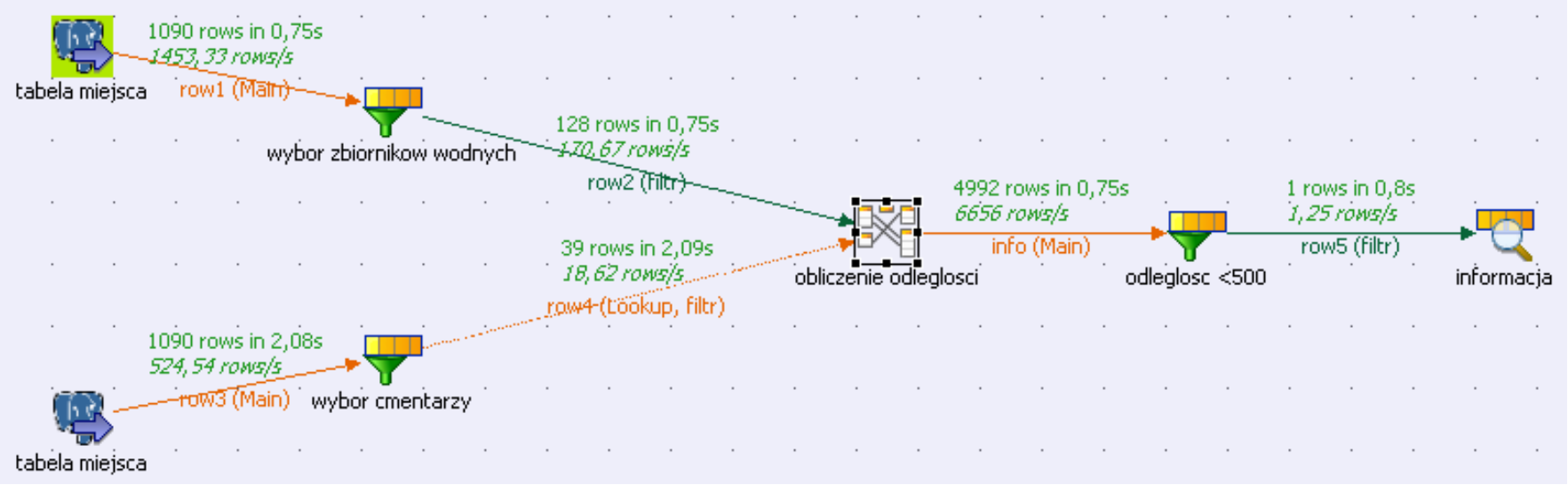


Jajeśnica Ł.,Piórkowski A. : Performance of Joins and Nesting for Normalized and Denormalized Schemas. Studia Informatica 2010, Vol. 31

EXAMPLES

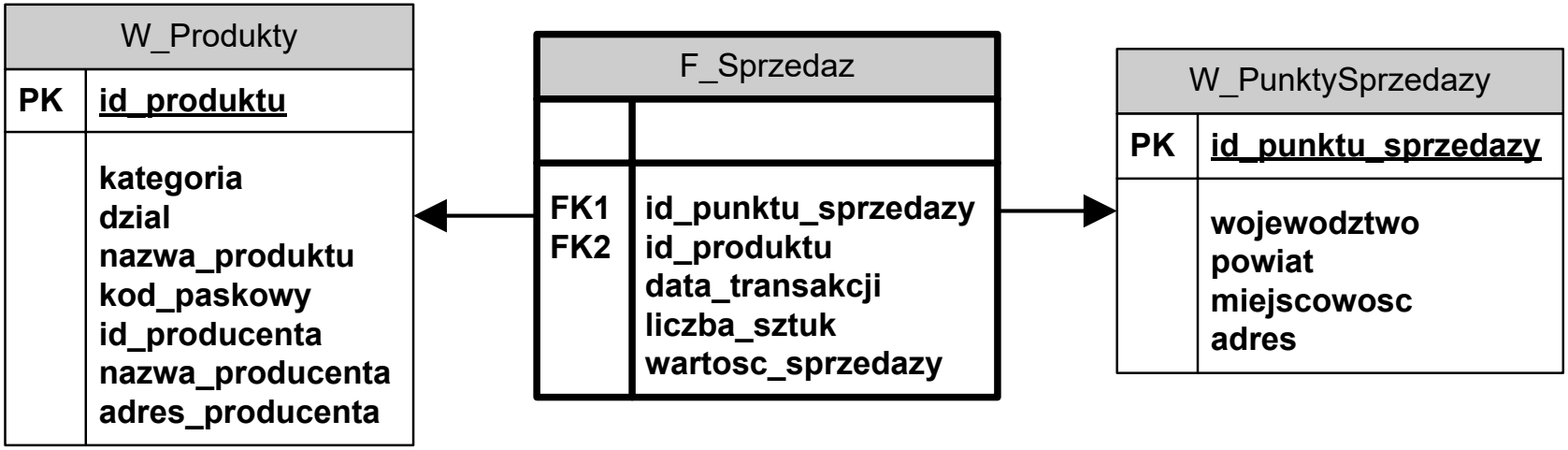

WWTP
PK id_wwtp
FK1 id_city
name_wwtp
hasdailydata
WWTPData
PK id_wwtpdata
FK2 id_wwtp
inflow
BOD
COD
carbon
nitrogen
phosphorus
accident
RLM
FK1 id_technology
date
WasteWaterTechnology
PK id_technology
name
WeatherData
PK id_weatherdata
FK1 id_weatherstation
date
mean_temperature
max_temperature
min_temperature
precipitation_amount
WeatherStations
PK id_weatherstation
FK1 id_city
station_name
longitude
latitude
altitude
Cities
PK id_city
city_name

EXAMPLES

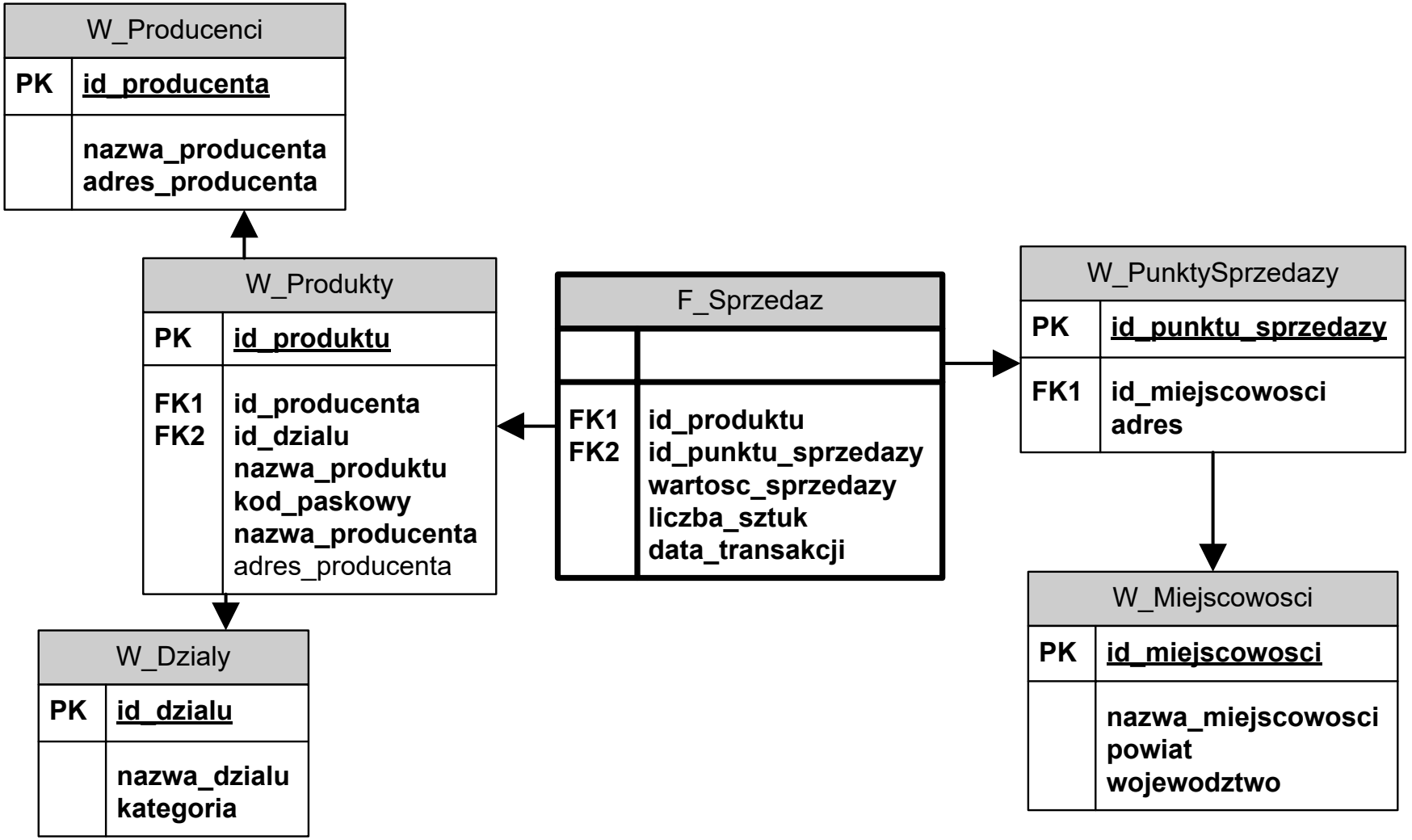


Klisiewicz, J., Piórkowski, A., Porzycka, S. (2011). Construction of an ETL Process for Spatial Data. Studia Informatica, 32(2B), 131-142.

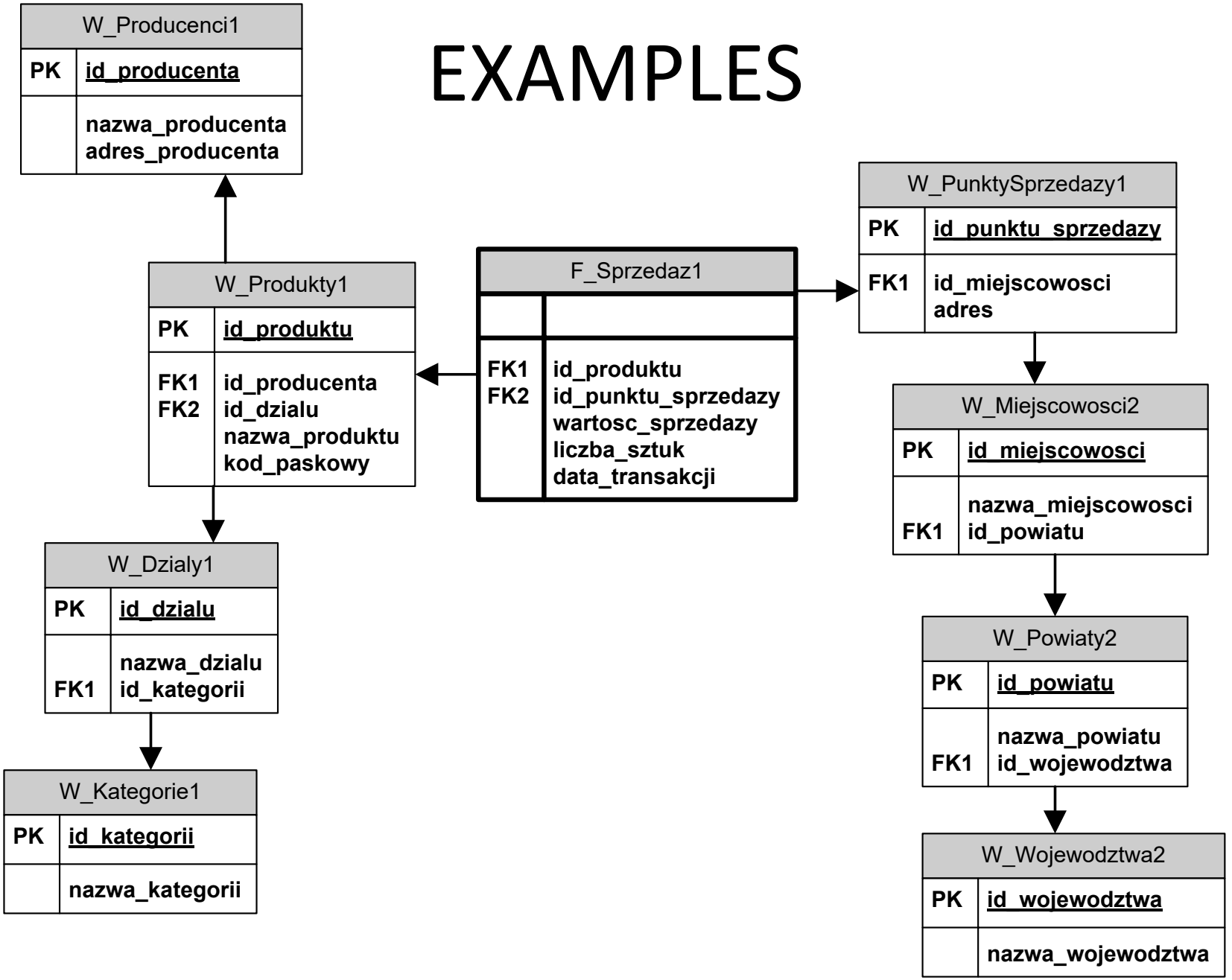
EXAMPLES


W_Produkty
PK
id_produktu
kategoria
dzial
nazwa_produktu
kod_paskowy
id_producenta
nazwa_producenta
adres_producenta
F_Sprzedaz
FK1
FK2
id_punktu_sprzedazy
id_produktu
data_transakcji
liczba_sztuk
wartosc_sprzedazy
W_PunktySprzedazy
PK
id_punktu_sprzedazy
wojewodztwo
powiat
miejscowosc
adres

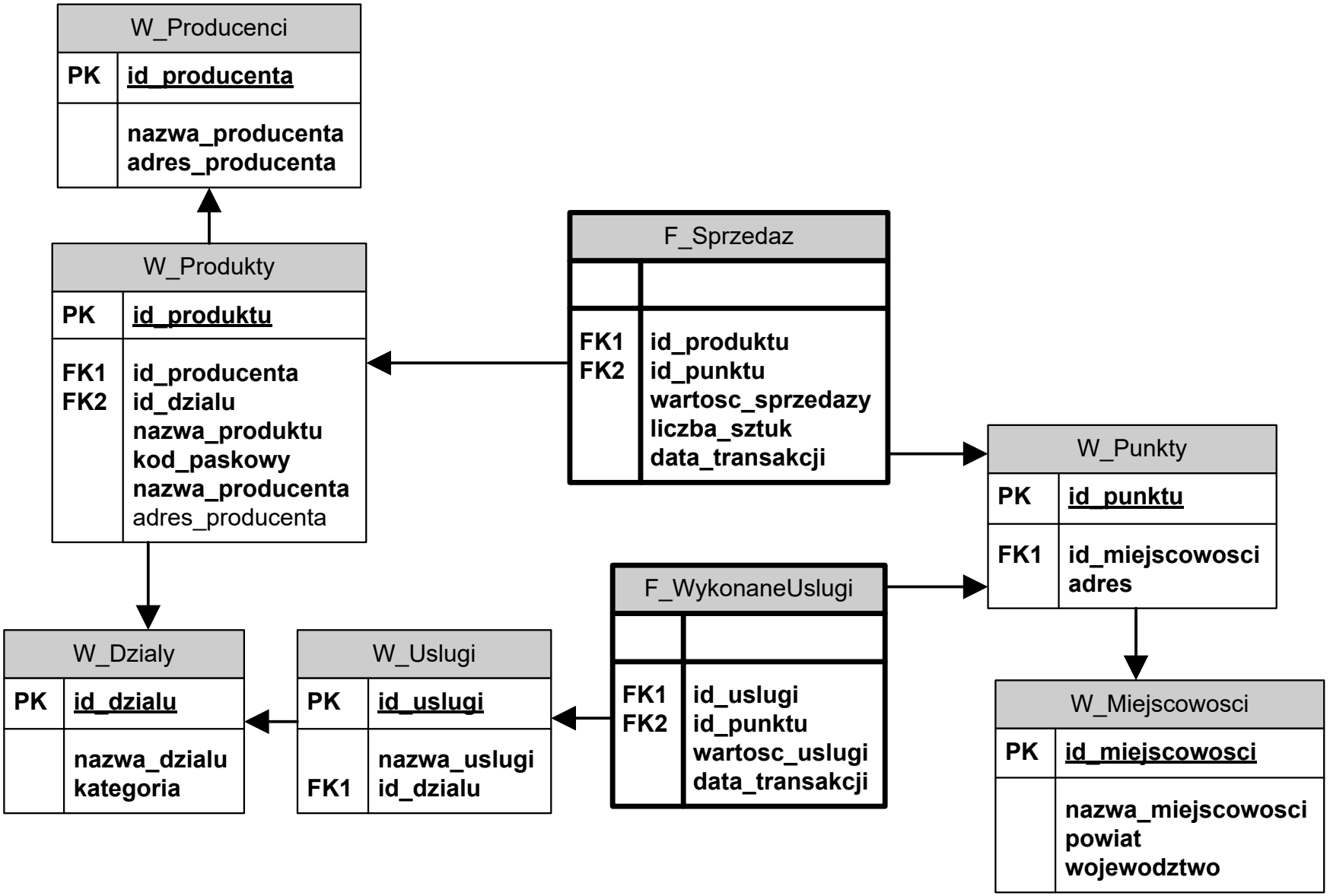
EXAMPLES


W_Producenci
PK
id_producenta
nazwa_producenta
adres_producenta
W_Produkty
PK
id_produktu
FK1
FK2
id_producenta
id_dzialu
nazwa_produktu
kod_paskowy
nazwa_producenta
adres_producenta
W_Dzialy
PK
id_dzialu
nazwa_dzialu
kategoria
F_Sprzedaz
FK1
FK2
id_produktu
id_punktu_sprzedazy
wartosc_sprzedazy
liczba_sztuk
data_transakcji
W_PunktySprzedazy
PK
id_punktu_sprzedazy
FK1
id_miejscowosci
adres
W_Miejscowosci
PK
id_miejscowosci
nazwa_miejscowosci
powiat
wojewodztwo

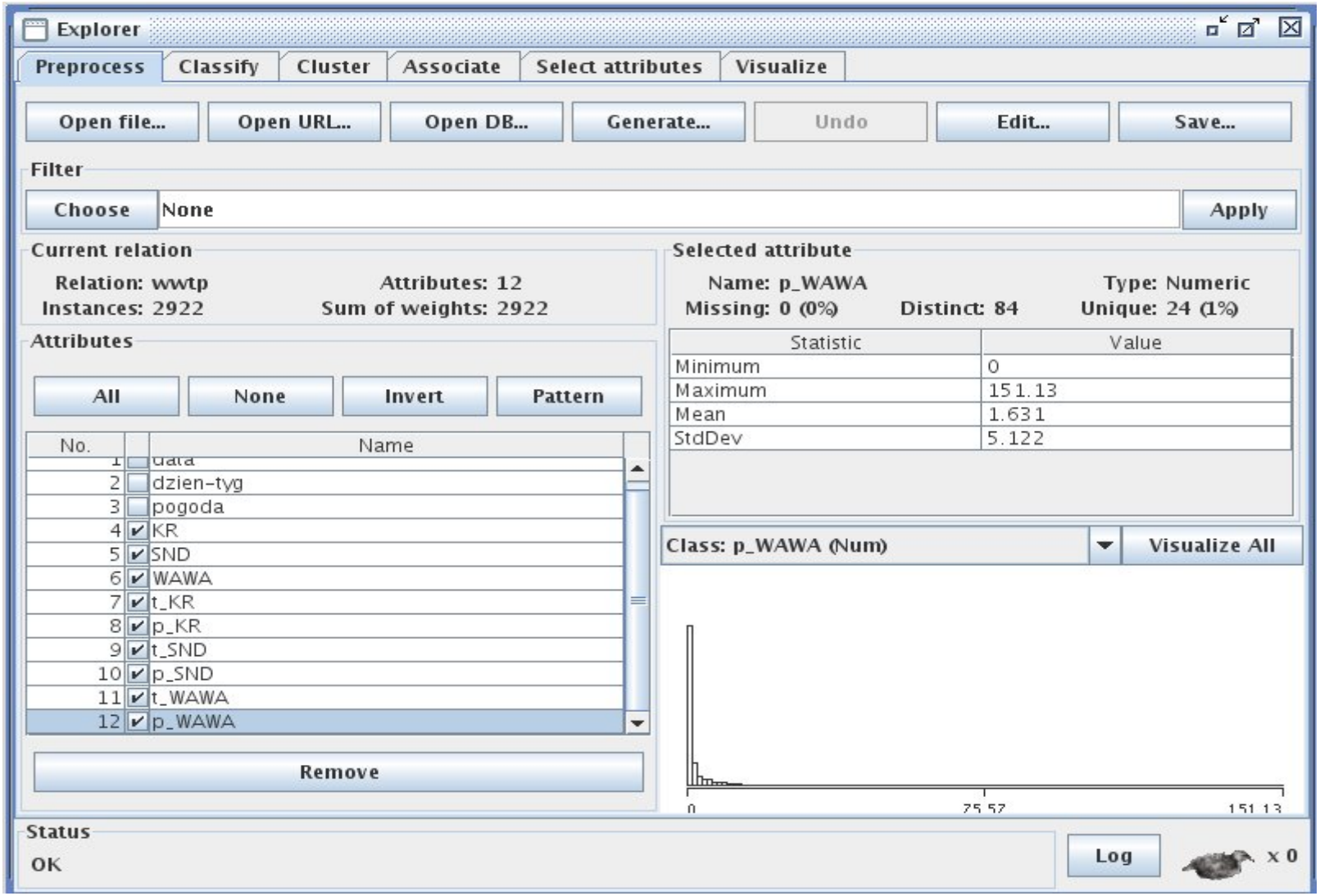
EXAMPLES


W_Producenci1
PK
id_producenta
nazwa_producenta
adres_producenta
W_Produkty1
PK
id_produktu
FK1
FK2
id_producenta
id_dzialu
nazwa_produktu
kod_paskowy
F_Sprzedaz1
FK1
FK2
id_produktu
id_punktu_sprzedazy
wartosc_sprzedazy
liczba_sztuk
data_transakcji
W_PunktySprzedazy1
PK
id_punktu_sprzedazy
FK1
id_miejscowosci
adres
W_Miejscowosci2
PK
id_miejscowosci
FK1
nazwa_miejscowosci
id_powiatu
W_Powiaty2
PK
id_powiatu
FK1
nazwa_powiatu
id_wojewodztwa
W_Wojewodztwa2
PK
id_wojewodztwa
nazwa_wojewodztwa
W_Dzialy1
PK
id_dzialu
FK1
nazwa_dzialu
id_kategorii
W_Kategorie1
PK
id_kategorii
nazwa_kategorii

EXAMPLES

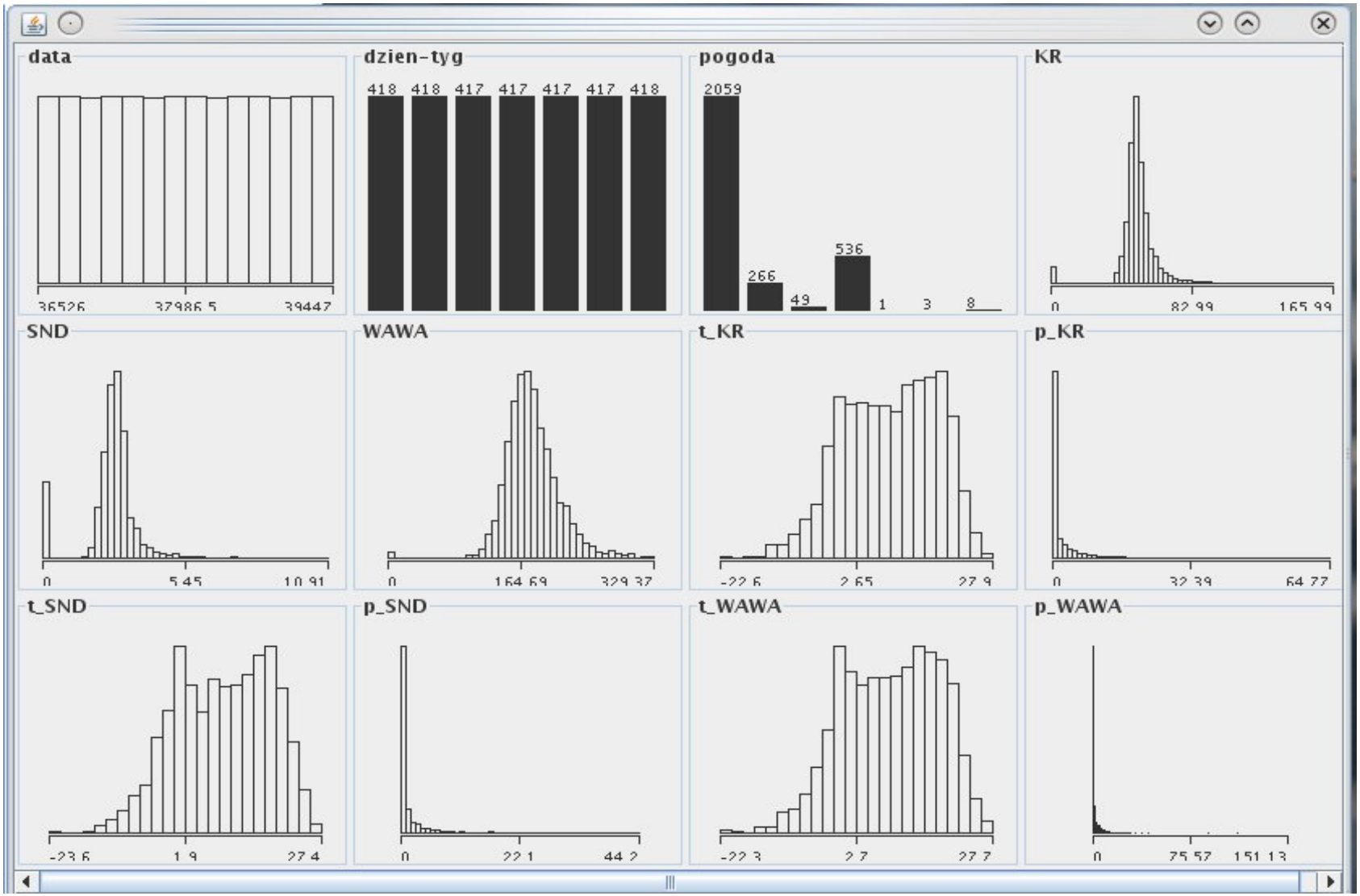

W_Producenci
PK
id_producenta
nazwa_producenta
adres_producenta
W_Produkty
PK
id_produktu
FK1
FK2
id_producenta
id_dzialu
nazwa_produktu
kod_paskowy
nazwa_producenta
adres_producenta
F_Sprzedaz
FK1
FK2
id_produktu
id_punktu
wartosc_sprzedazy
liczba_sztuk
data_transakcji
W_Punkty
PK
id_punktu
FK1
id_miejscowosci
adres
W_Dzialy
PK
id_dzialu
nazwa_dzialu
kategoria
W_Uslugi
PK
id_uslugi
FK1
nazwa_uslugi
id_dzialu
F_WykonaneUslugi
FK1
FK2
id_uslugi
id_punktu
wartosc_uslugi
data_transakcji
W_Miejscowosci
PK
id_miejscowosci
nazwa_miejscowosci
powiat
wojewodztwo

EXAMPLES



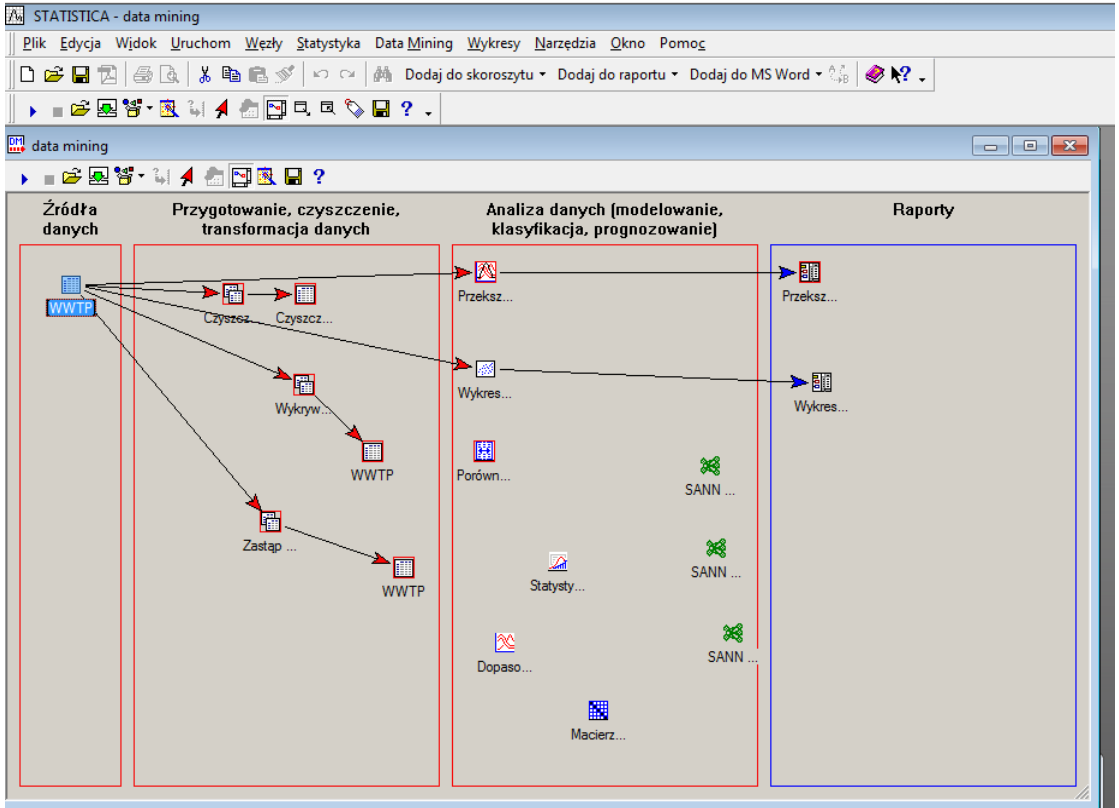
M. Chuchro.,Piórkowski A. : Use of Data Mining Methods and Tools for Analysing Variability in Inflow to Municipal Wastewater Treatment Plants. Studia Informatica 2010, Vol. 31

EXAMPLES



M. Chuchro.,Piórkowski A. : Use of Data Mining Methods and Tools for Analysing Variability in Inflow to Municipal Wastewater Treatment Plants. Studia Informatica 2010, Vol. 31

EXAMPLES

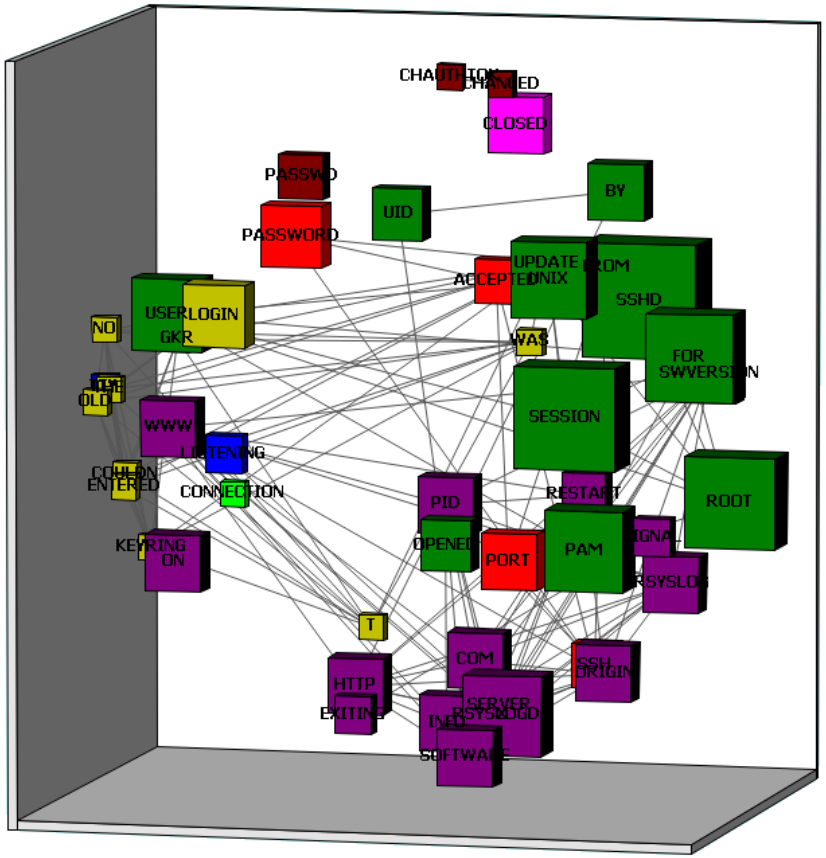


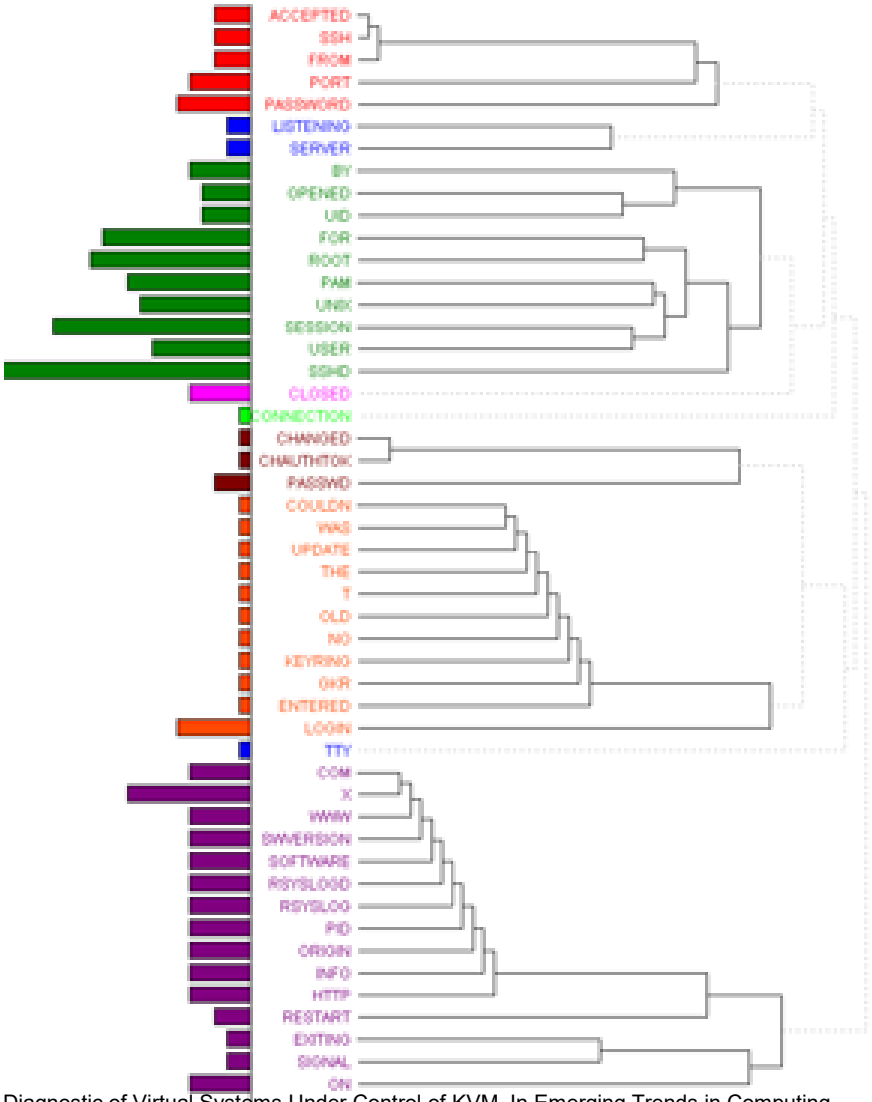
	1 Data	2 Data	3 Pogoda	4 Kraków - Kujawy	5 Sandomierz	6 Warszawa - Czajka	7 T Kr	8 PP Kr	9 T Snd	10 PP Snd	11 T Wawa	PF
1	36526	7	s	43	2,587	167,215	-8,6	0	-3,8	0	-1,1	
2	36527	1	s	43,6	2,621	171,455	-5,9	0	-8	0	-5,7	
3	36528	2	s	40,8	2,83	187,2	1,2	0,76	-0,3	0,76	1,2	
4	36529	3	od	58	2,9	223,55	2,3	0,76	0,8	1,02	2,3	
5	36530	4	od	58,2	2,95	225,34	2,8	2,03	1,2	2,54	2,7	
6	36531	5	od	51,3	3,05	193,5	0,4	0	0,7	0,25	0,8	
7	36532	6	s	46,7	2,816	193,85	-1,7	0	-0,8	0	0,4	
8	36533	7	s	54,2	3,05	187,99	0,3	0	-0,2	0	1,8	
9	36534	1	s	50,2	2,86	179,8	0,9	0	-0,3	0	1,2	
10	36535	2	s	51,1	3,001	183,91	0,9	0,76	0,6	0	1,6	
11	36536	3	s	50,8	2,99	190,37	0,2	0,76	0,2	0	0,6	

Gotowe

M. Chuchro.,Piórkowski A. : Use of Data Mining Methods and Tools for Analysing Variability in Inflow to Municipal Wastewater Treatment Plants. Studia Informatica 2010, Vol. 31

EXAMPLES





Chuchro, M., Szostek, K., Piórkowski, A., & Danek, T. (2013). Using Data Mining Techniques for Diagnostic of Virtual Systems Under Control of KVM. In Emerging Trends in Computing, Informatics, Systems Sciences, and Engineering (pp. 1011-1022). Springer New York.

DESIGN OF (RELATIONAL) DATABASES

Representation of a relation (table)

Relational constraints

ERD diagrams

* Chen Notation
* Martin Notation

Schema normalization

* anomalies in relational databases
* normal forms

Relation (Table)

Relation schema = relation name + set of attributes

Studenci(imie, last name, PESEL, student_id)

Akademiki(nr_ds, university, city, street, number)

SaleWykladowe(building, room_no, pietro, number_of_seats, projector)

Table Design

- Table name – set name – plural form?
- Primary key
- Set of attributes – data types

Data Types

TYP	[B]	ZAKRES	OPIS
INTEGER, INT, BIGINT, SMALLINT, TINYINT	4 8,2,1	-2^{31} do 2^{31} -1 -2,147,483,648 - 2,147,483,647	Liczby całkowite
DECIMAL (precyzja, skala) NUMERIC*	~ precyzji	max liczba cyfr ~SZBD	Liczby dziesiętne, np. waluta: 100.15
FLOAT, REAL	Precyzja 1-24 / 25-53 4	-1.79E+308 :-2.23E-308, 2.23E-308 to 1.79E+308 - 3.40E + 38 to -1.18E – 38 1.18E - 38 to 3.40E + 38.	Liczby zmiennoprzecinkowe

In SQL, REAL corresponds to the Intel processor float type (4 bytes), whereas FLOAT corresponds to double precision

NUMERIC has fixed precision, DECIMAL may have greater precision than required by the designer (related to DBMS performance

SQL Standard

Data Types

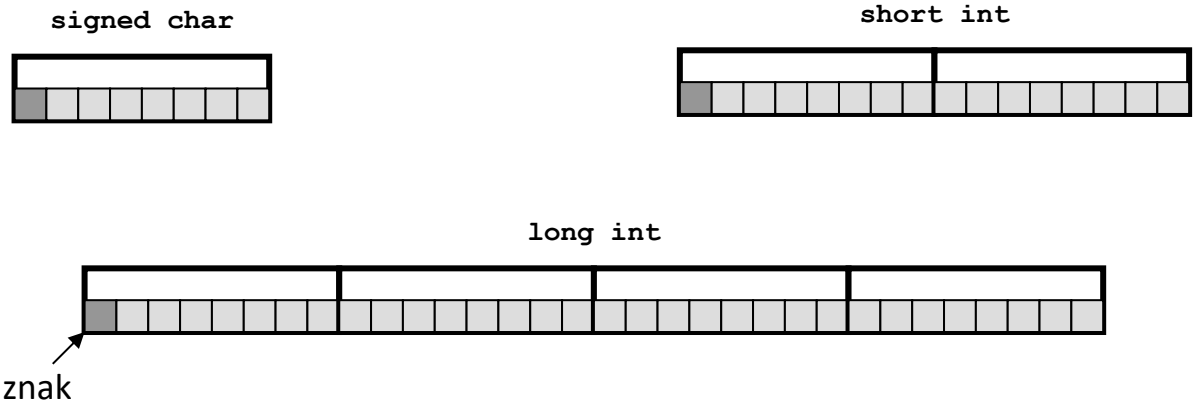
TYP	[B]	ZAKRES	OPIS
CHAR CHAR(20)	1 20	Jeden znak ASCII Tablica 20 znaków ASCII	Jeden znak Dokładnie 20 znaków, dopełniane spacjami
VARCHAR(_X_)	1B - 2GB	Tablica znaków ASCII	Dowolna liczba znaków, ograniczona przez _X_
NVARCHAR(_X_)	2B - 2GB	Tablica znaków UNICODE	Można składować tekst w znakach narodowych
DATE, TIME, DATETIME	10B,..	Np. tekstowo YYYY-MM-DD	czas
BLOB CLOB	Np. do 2GB	Blok binarny Blok tekstowy	Np. na zdjęcia Np. na strony

SQL Standard

Data Types

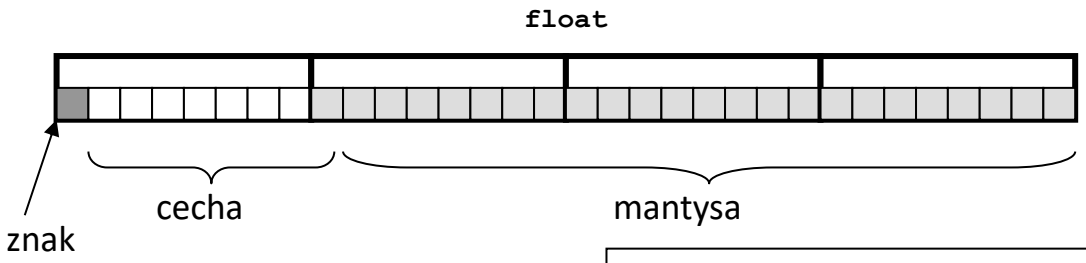
integers

signed



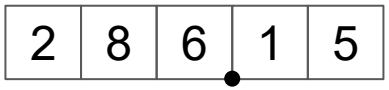
floating-point numbers

signed



number = mantissa × 2^exponent

decimal numbers
(fixed precision)

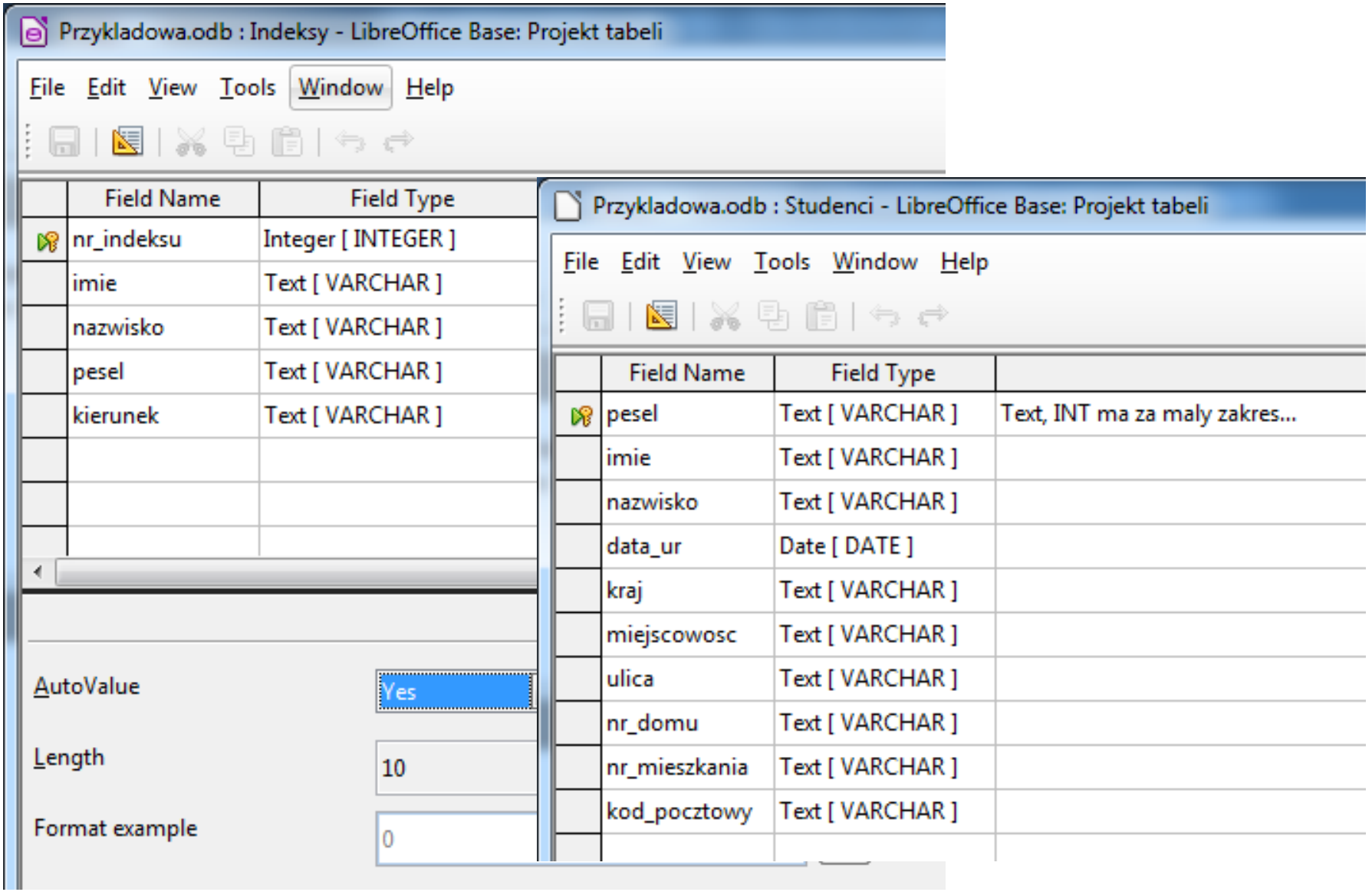


Floating-Point Numbers

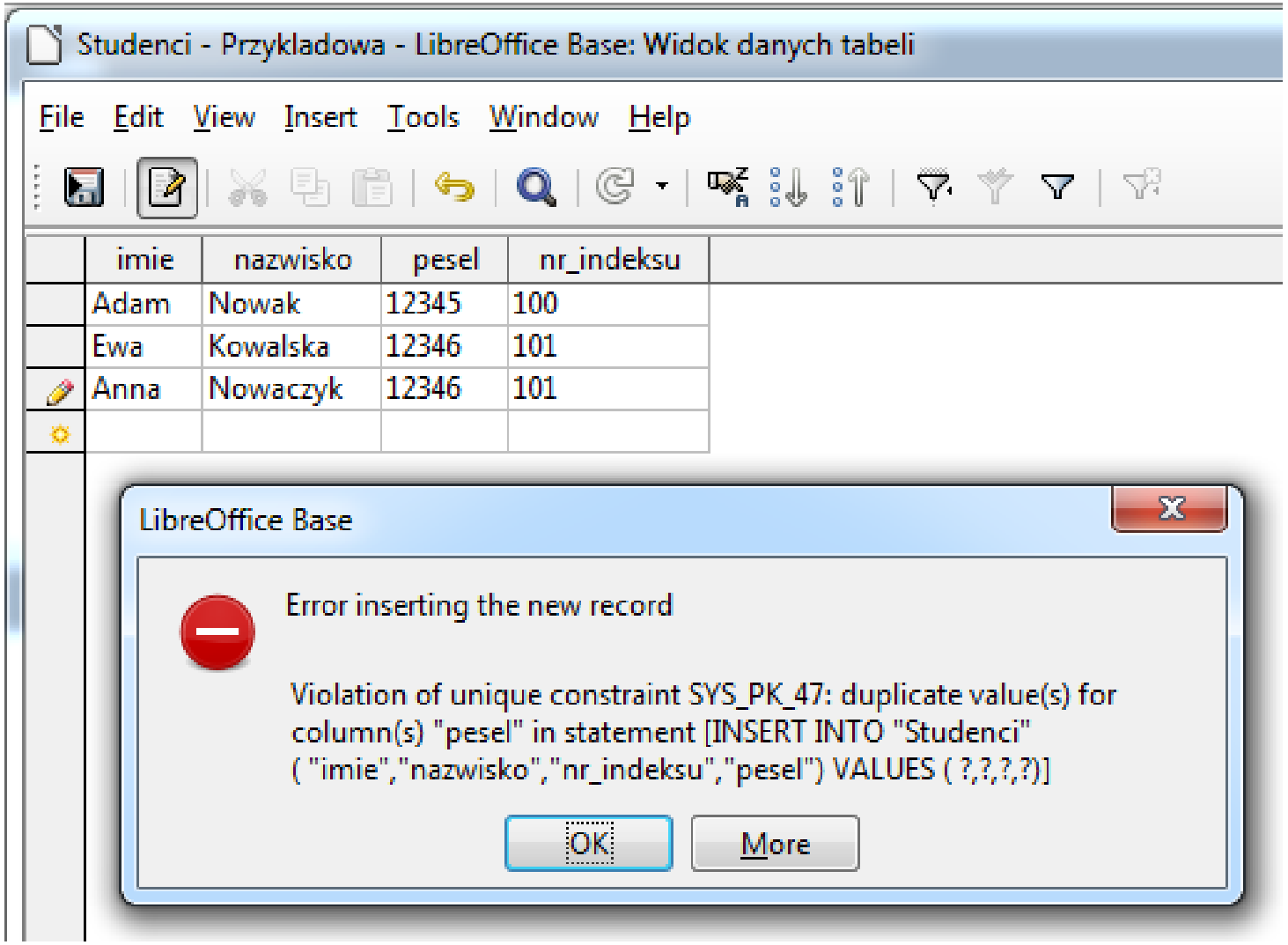
```
CREATE TABLE liczby(i INT, b BIGINT, f FLOAT, r REAL);

INSERT INTO liczby VALUES
     ( 1234567891, 1234567891234567891, 123456789123456789, 123456789123456789 );

SELECT * FROM liczby;
     i      |          b          |          f           |      r_x000B_------------+--------
-------------+----------------------+-------------_x000B_ 1234567891 | 1234567891234567891 |
1.23456789123457e+017 | 1.23457e+017



DELETE FROM liczby;

INSERT INTO liczby VALUES( 0, 0,  100000,  100000 );_x000B_INSERT INTO liczby VALUES( 0, 0,
1000000, 1000000 );_x000B_SELECT * FROM liczby;


 i | b |    f    |   r_x000B_---+---+---------+--------_x000B_ 0 | 0 |  100000 | 100000_x000B_
0 | 0 | 1000000 | 1e+006



UPDATE liczby SET f = f + 0.007;_x000B_UPDATE liczby SET r = r + 0.007;

SELECT * FROM liczby;
 i | b |      f      |   r_x000B_---+---+-------------+--------_x000B_ 0 | 0 |  100000.007 |
100000_x000B_ 0 | 0 | 1000000.007 | 1e+006
```

Indexes(nr_indeksu, imie, nazwisko, PESEL, kierunek)

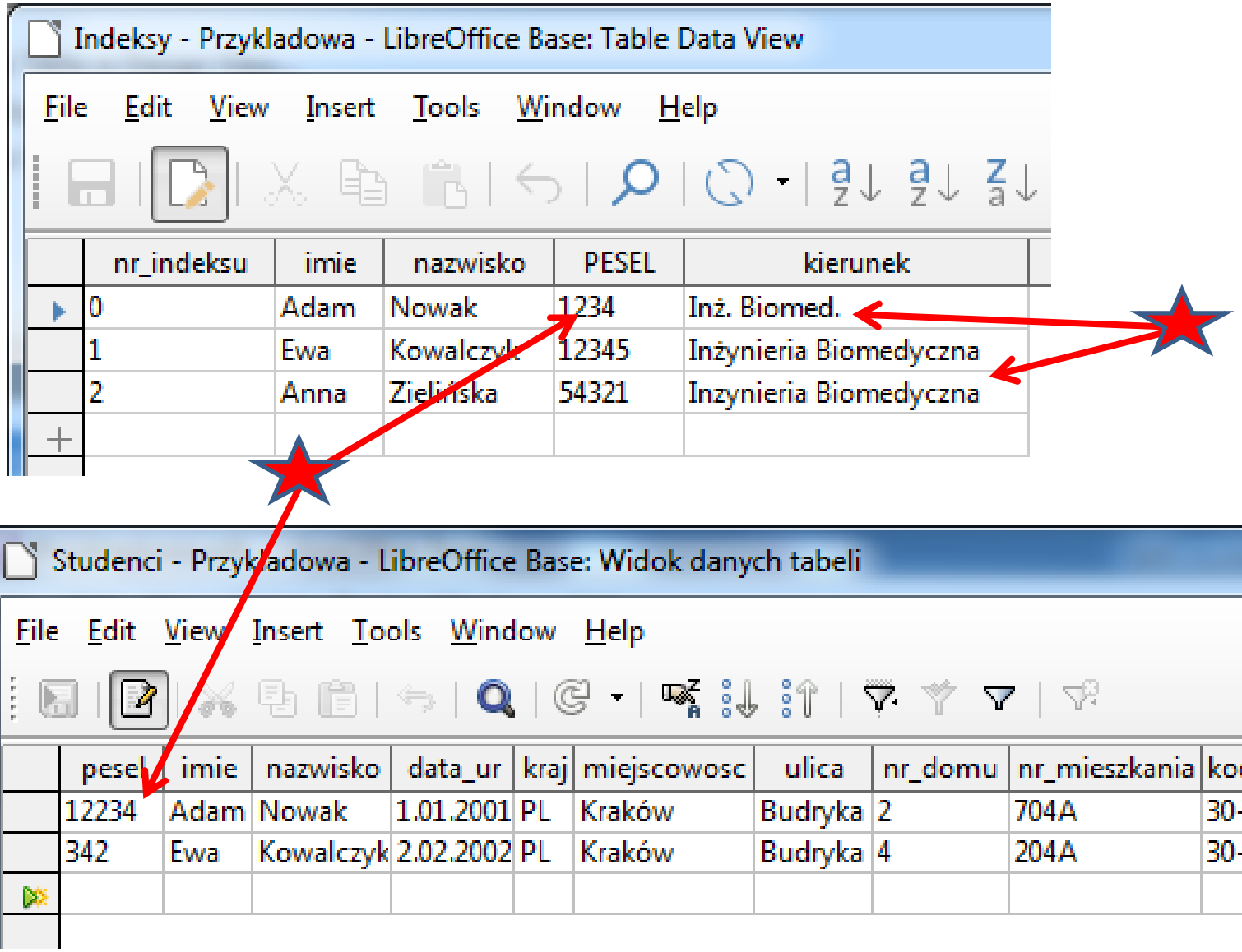
Table Design



Entering Values



Entering Values



ENTITY RELATIONSHIPS, RELATIONAL CONSTRAINTS

- Enforcement of data integrity between two tables.

Relationships may exist between two entity sets A and B. Assume that A is the source set and B is the target set.

- Relationships are unidirectional (although reverse relationships may also exist).
- A mapping is total if every entity in the source set has a corresponding image.
- A mapping is partial if some entities in the source set have no corresponding image.

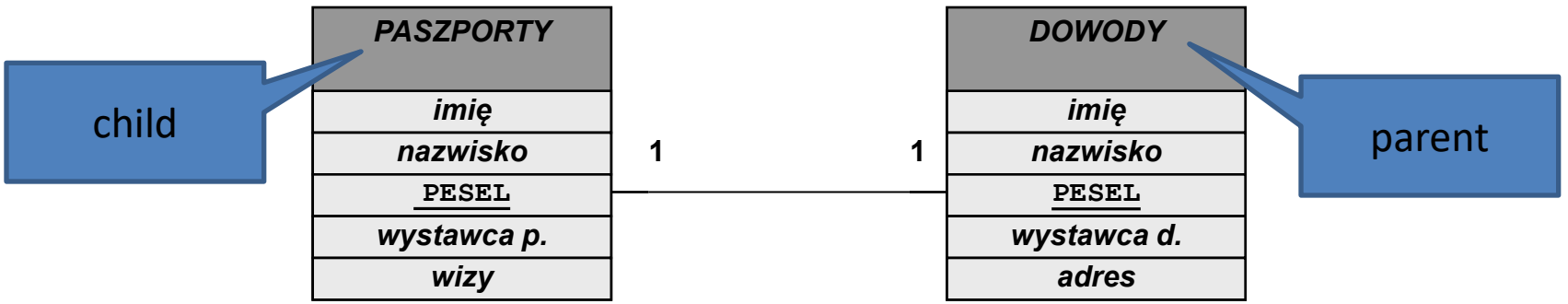
- Relationship cardinalities: 1:1, 1:N and M:N (!)

1:1 – one-to-one

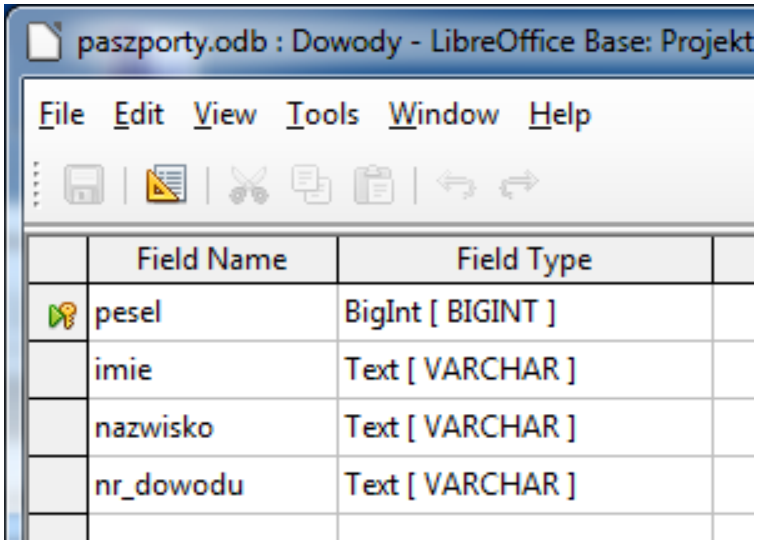
In the child table, the foreign-key (FK) column may contain only values from the specified column (e.g. the primary key) of the parent table, and these values must not be duplicated.

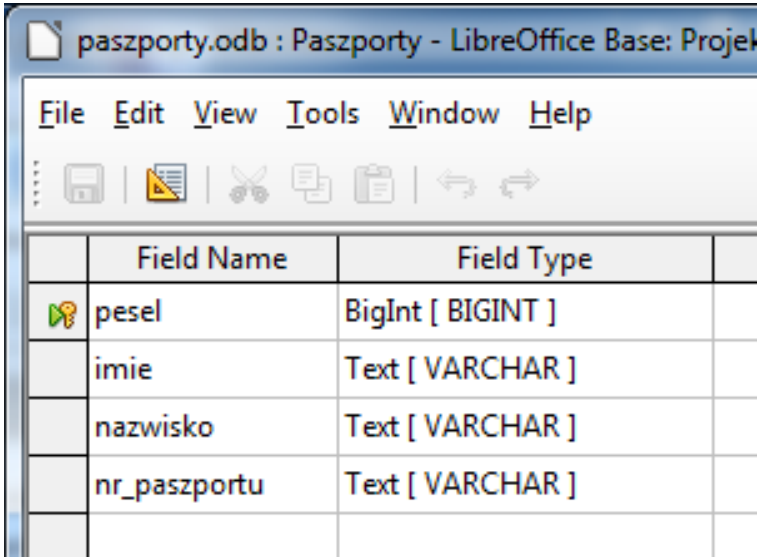
example:
Passports(first name, last name, PESEL, passport issuer, visas)
Identity_Cards(first name, last name, PESEL, id_card_issuer, residential_address).

Assumptions:
a)everyone must have an identity card, but only selected individuals have a passport (this does not correspond, for example, to the situation of foreign students);
b)only one identity card and one passport may be held;
c)conclusion: a passport can be issued only for an existing identity card.

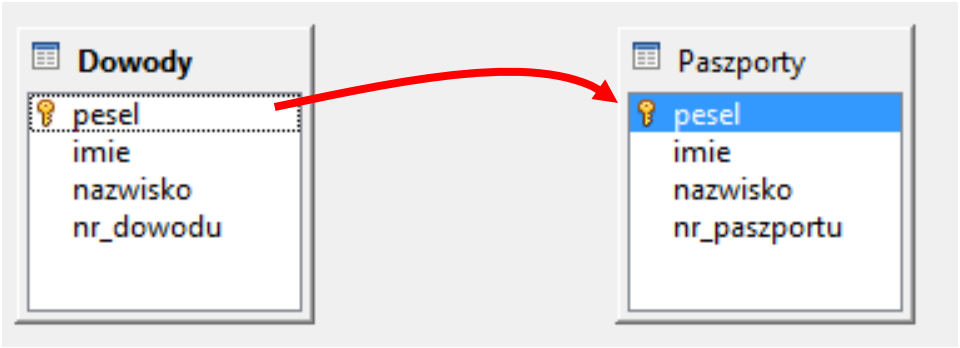


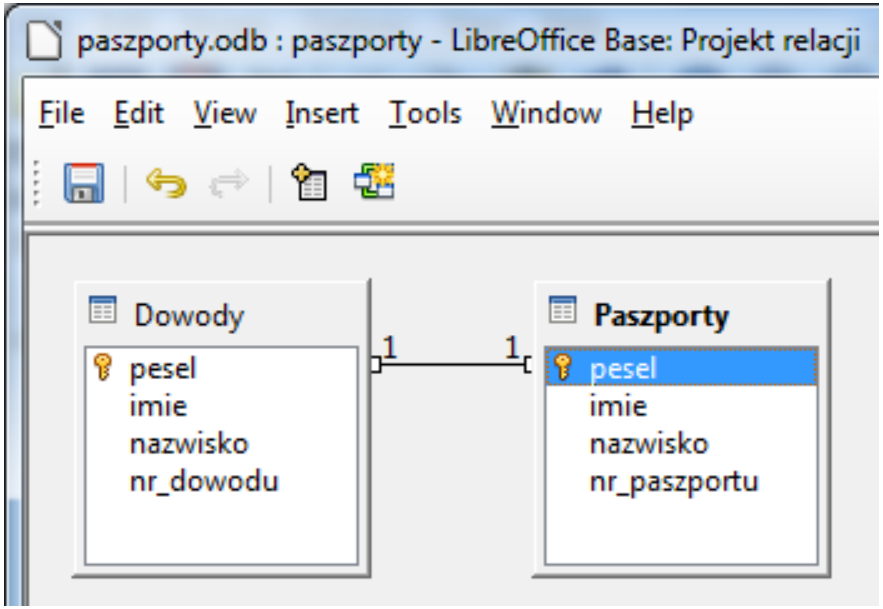
1:1 – one-to-one



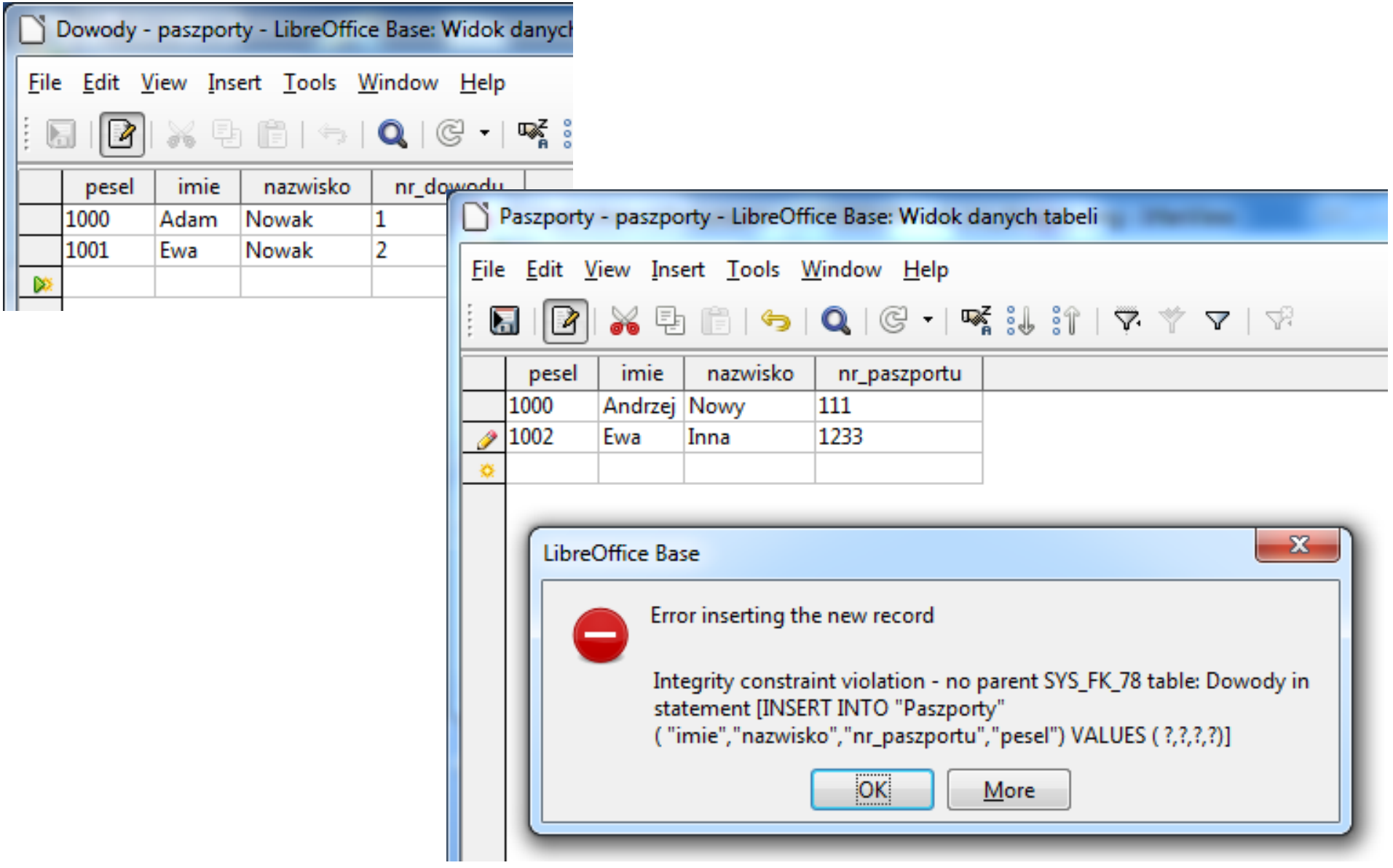


LibreOffice: drag the parent onto the child





1:1 – one-to-one

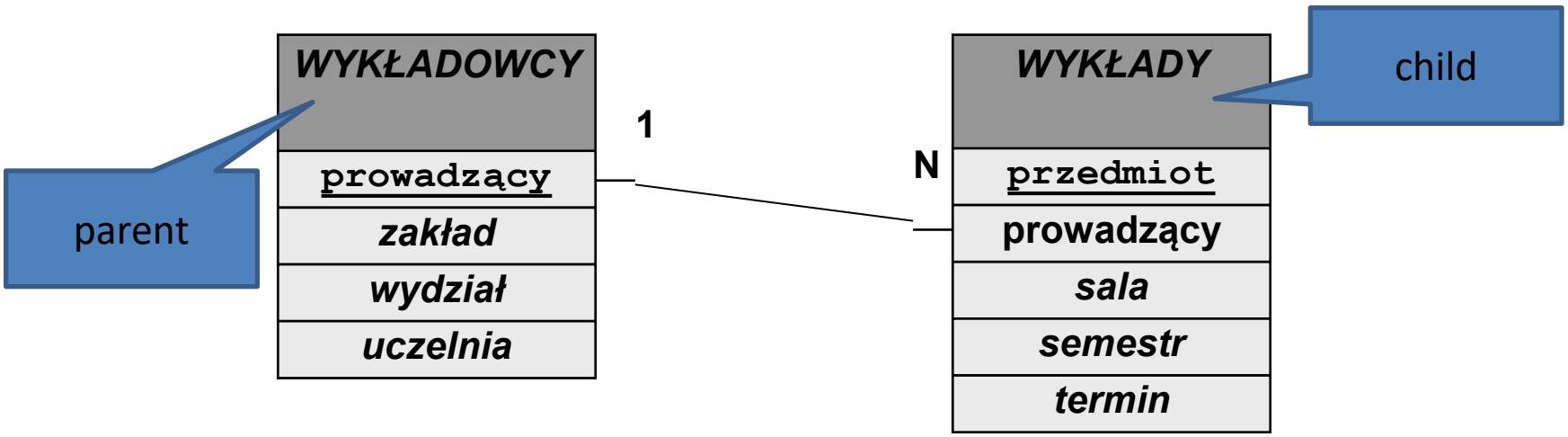

Dowody - paszporty - LibreOffice Base: Widok danych
File Edit View Insert Tools Window Help
pesel imie nazwisko nr_dowodu
1000 Adam Nowak 1
1001 Ewa Nowak 2
Paszporty - paszporty - LibreOffice Base: Widok danych tabeli
File Edit View Insert Tools Window Help
pesel imie nazwisko nr_paszportu
1000 Andrzej Nowy 111
1002 Ewa Inna 1233
LibreOffice Base
Error inserting the new record
Integrity constraint violation - no parent SYS_FK_78 table: Dowody in statement [INSERT INTO "Paszporty"
("imie","nazwisko","nr_paszportu","pesel") VALUES (?,?,?,?)]
OK
More

1:N – one-to-many

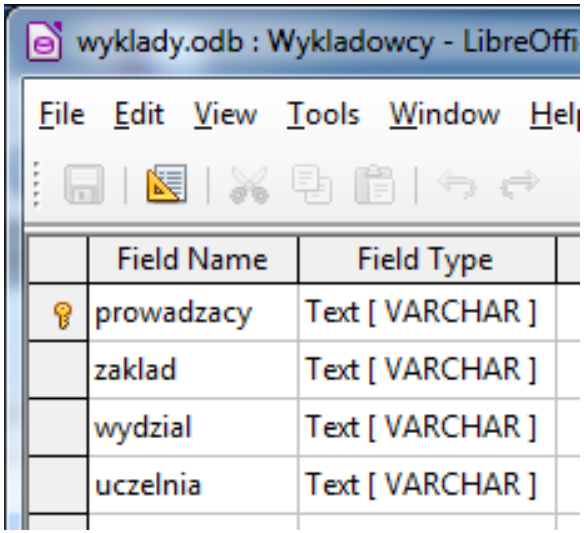
In the child table, the foreign-key (FK) column may contain only values from the specified column (e.g. the primary key, PK) of the parent table. These values may be repeated (N), whereas the corresponding primary-key values in the parent table are unique.

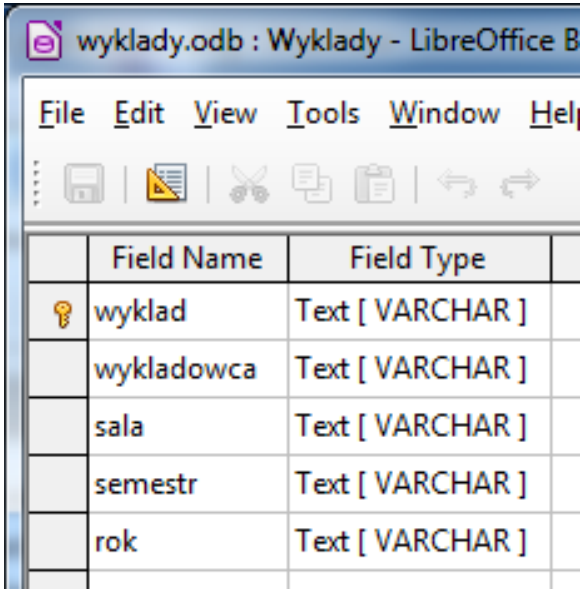
example:
lecturers: lecturer, department, faculty, university
lectures: course, lecturer, room, semester, date/time

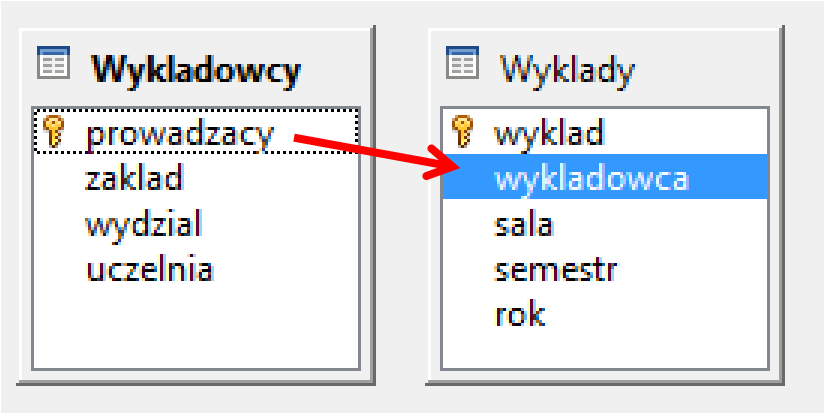
Assumptions:
a)each lecture must be taught by one existing lecturer;
b)each lecturer may teach 0–N lectures.

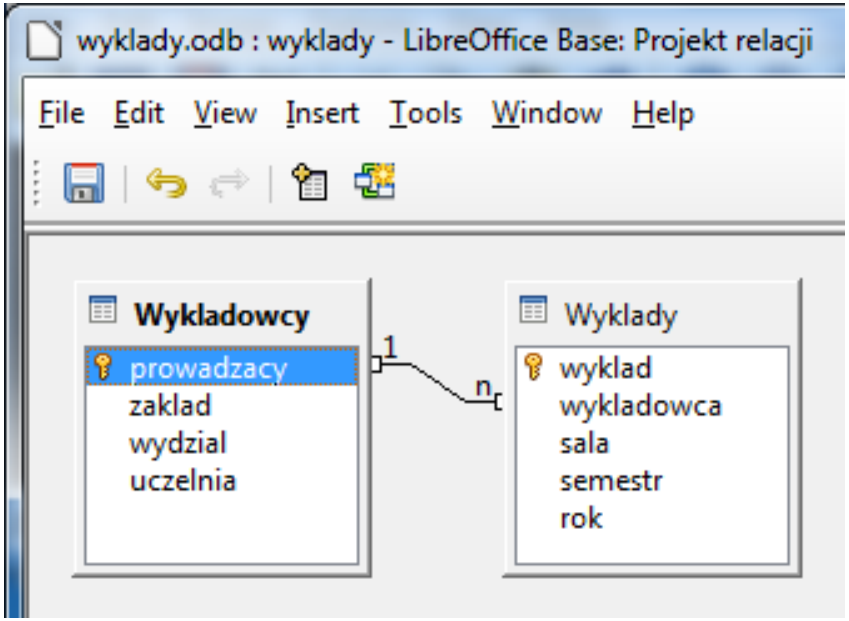


1:N – one-to-many



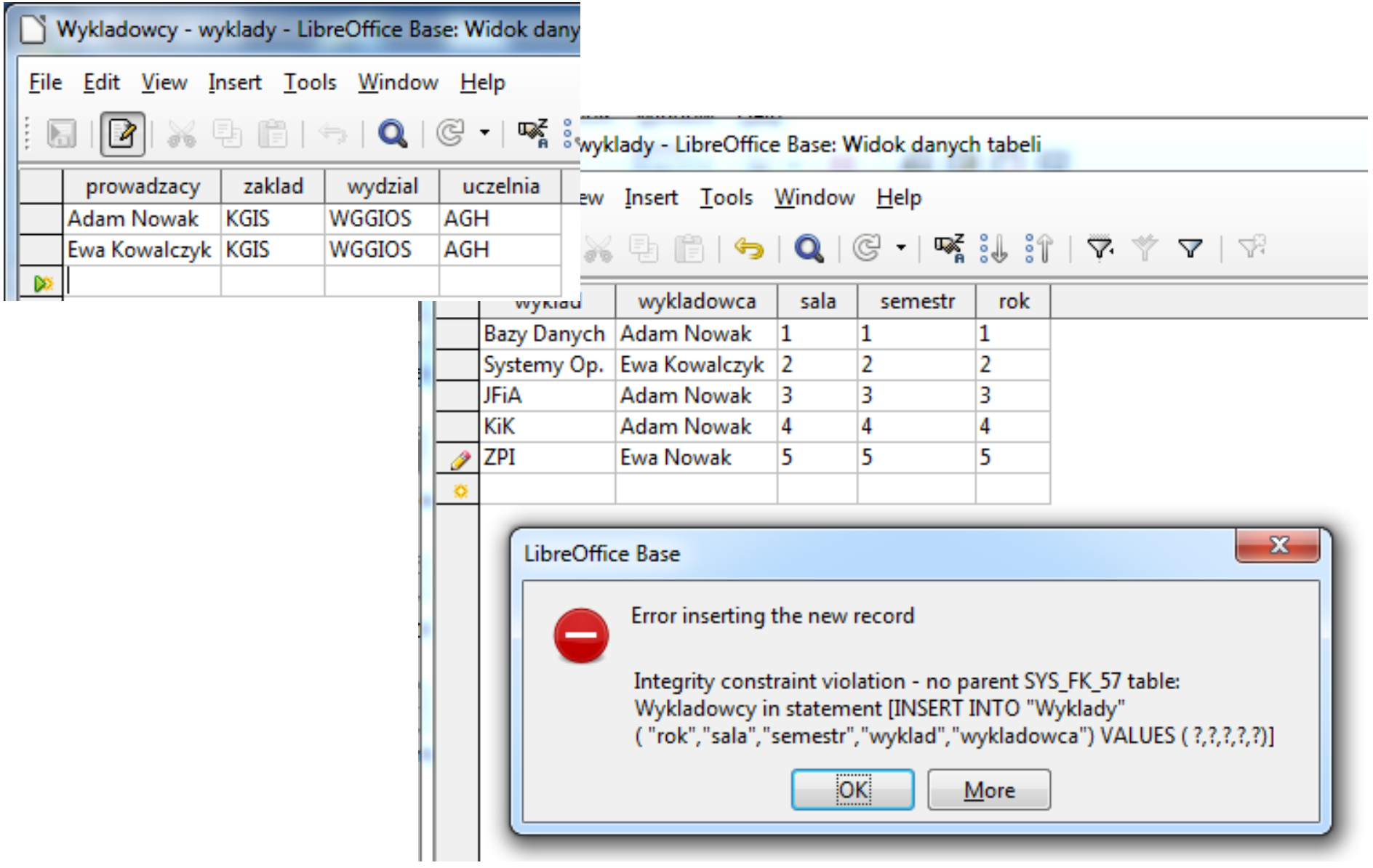






The names of the attributes on which the relationship is based may, but need not, be identical

1:N – one-to-many

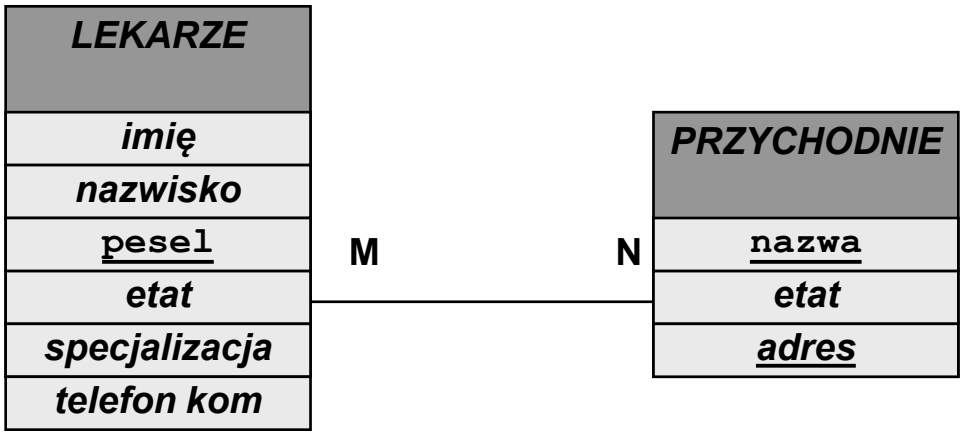


M:N – many-to-many

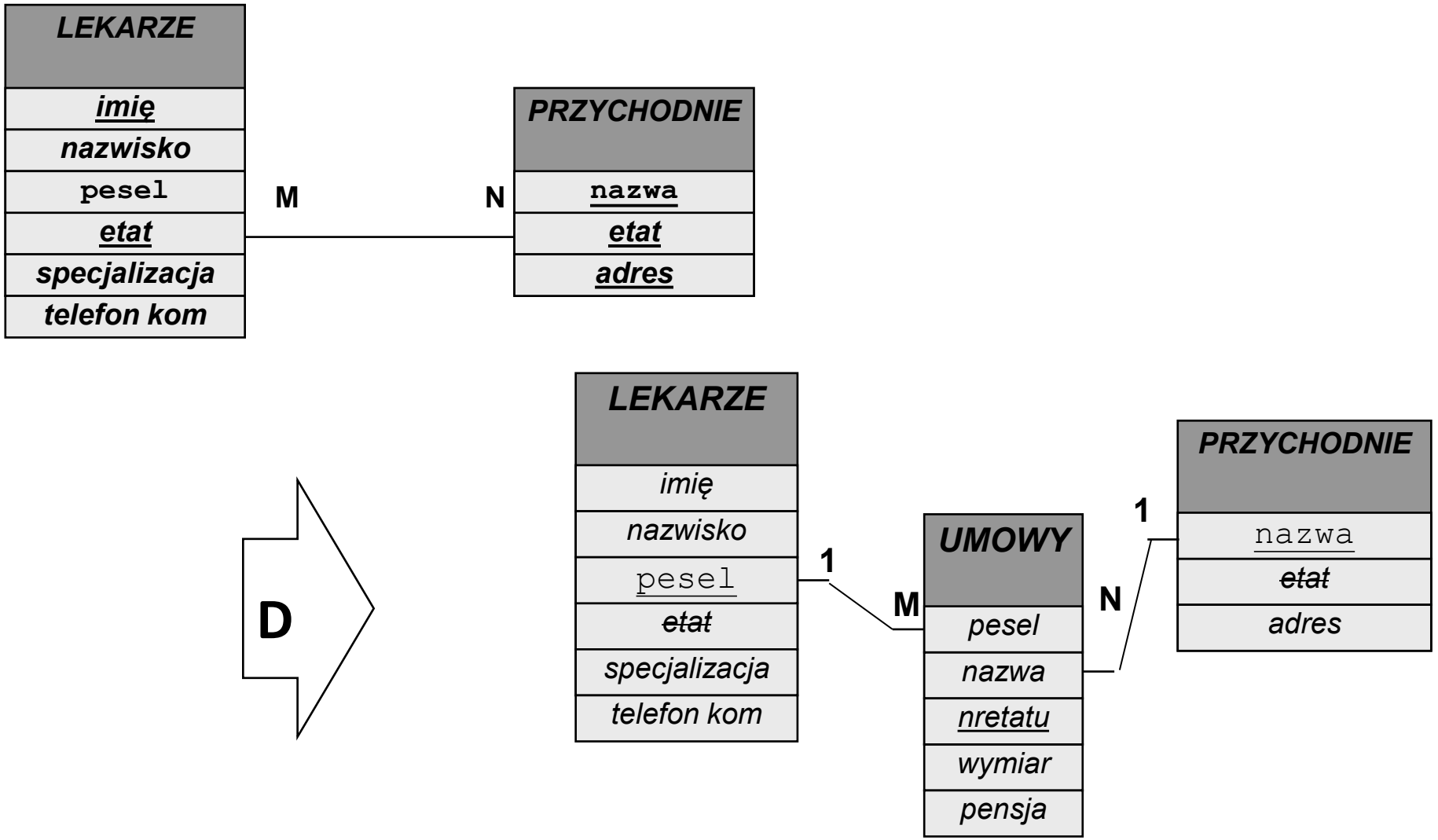
For a given key value, a subset of values in one table corresponds to a subset of values in the other table, and vice versa.
Such relationships cannot be implemented directly in a relational database and M:N must therefore be decomposed into two relationships, M:1 and 1:N.

example:
physicians: first name, last name, specialization, employment
clinics: name, address, employment

Assumption: each physician may work at different clinics, and clinics may employ different physicians.



many_x000B_decomposition of M:N into 1:M and N:1



Data Integrity Constraints

Wybrane województwo: **małopolskie**
Miejscowość: **Kraków**
Wybrany typ szkoły: **Szkoła wyższa (zweryfikowane)**

« poprzednie | 1 | 2 | następne »

Nazwa szkoły

Uniwersytet Jagielloński
Uniwersytet Jagielloński - Wydział Polonistyki
Uniwersytet Papieski Jana Pawła II
Uniwersytet Papieski Jana Pawła II
Uniwersytet Papieski Jana Pawła II
Uniwersytet Pedagogiczny im. Komisji Edukacji Narodowej (dawniej Akademia Peda
Uniwersytet Rolniczy
Wydział Hodowli i Biologii Zwierząt Akademii Rolniczej
Wydział Pedagogiczny - Uniwersytet Pedagogiczny
Wydział Prawa i Administracji - Studia Podyplowe-Uniwersytet Jagielloński

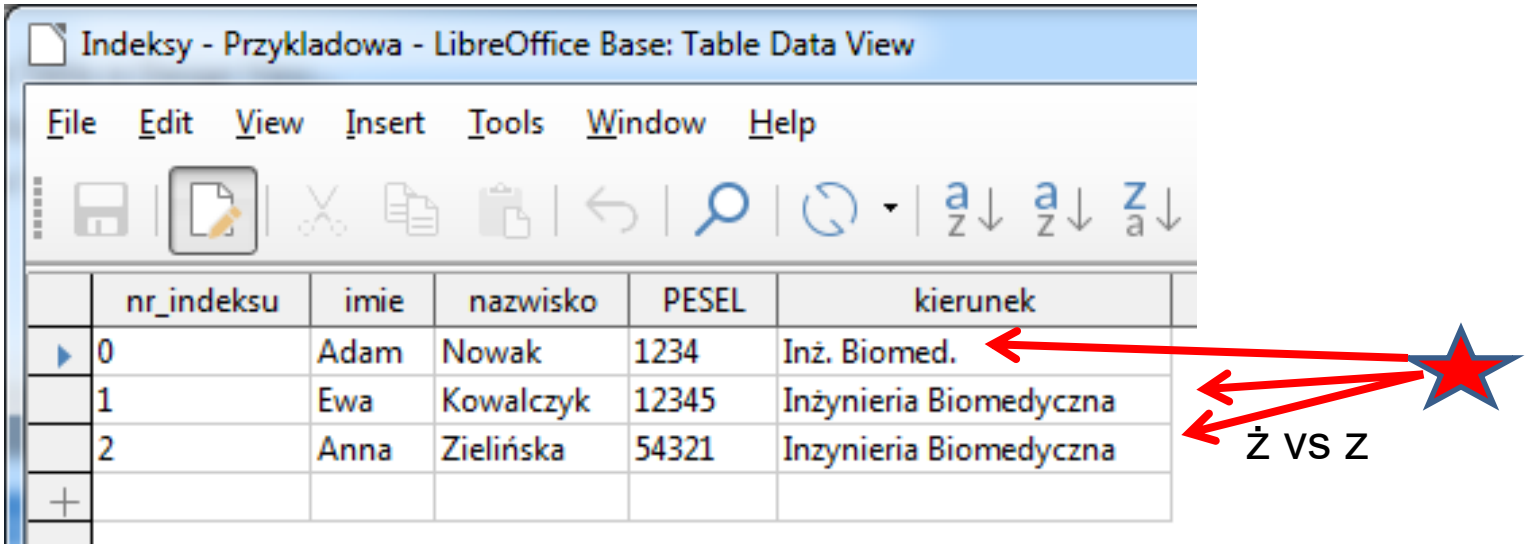
WSZYSTKIE KIERUNKI

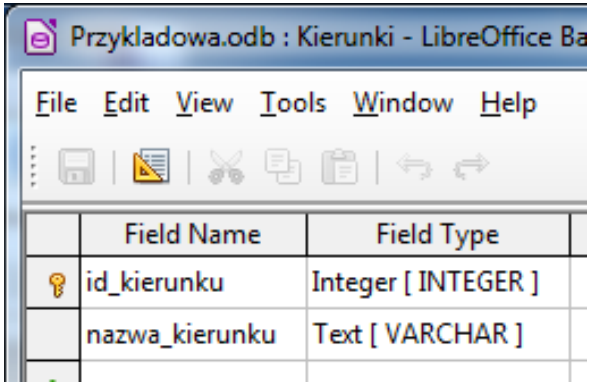
« poprzednie | 1 | 2 | 3 | 4 | 5 | ... | 166 | następne »

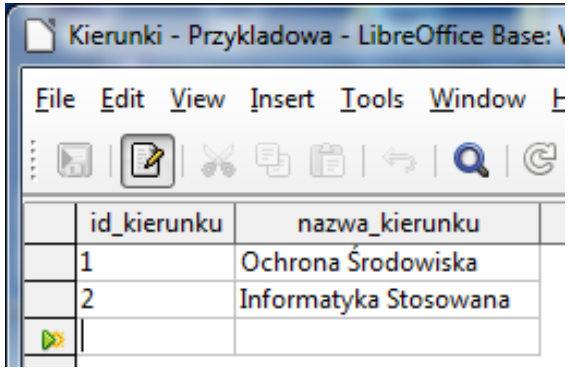
21.	Administracja	dzienne lic, 3–letnie	W. Prawa i Administracji
22.	Administracja	uzupelniajace zaoczne, 2–letnie	WPiA
23.	Administracja	zaoczne, 5–letnie	WPiA
24.	Administracja	zaoczne, 2–letnie	Prawa i Administracji
25.	Administracja	niestacjonarne, mgr, 5–letnie	Wydział Prawa i Administracji
26.	Administracja	Niestacjonarne-uzupełniające, 2–letnie	Prawa i Administracji
27.	Administracja	zaoczne, 3–letnie	Prawa i Administracji
28.	Administracja	zaoczne mgr, 2–letnie	Prawo i Administracja
29.	Administracja	zaoczne, 5–letnie	Prawo i Administracja
30.	Administracja	niestacjonarne licencjackie, 3–letnie	Prawa i Administracji
31.	Administracja	zaoczne, 5–letnie	Prawa i Administracji
32.	administracja	zaoczne, 5–letnie	Prawo i administracja
33.	Administracja	licencjackie, 3–letnie	Prawa i Administracji
34.	Administracja	dzienne mgr, 4–letnie	Prawo i Administracja
35.	Administracja	dzienne lic, 3–letnie	Prawa i Administracji
36.	Administracja	jednolite magisterskie, 5–letnie	Prawa i Administracji
37.	Administracja	zaoczne, 5–letnie	Wydział Prawa i Administracji
38.	Administracja	niestacjonarne lic, 3–letnie	Prawa i Administracji
39.	Administracja	SUM, 2–letnie	Prawa i Administarcji
40.	Administracja	zaoczne, 2–letnie	Prawa i Administracji

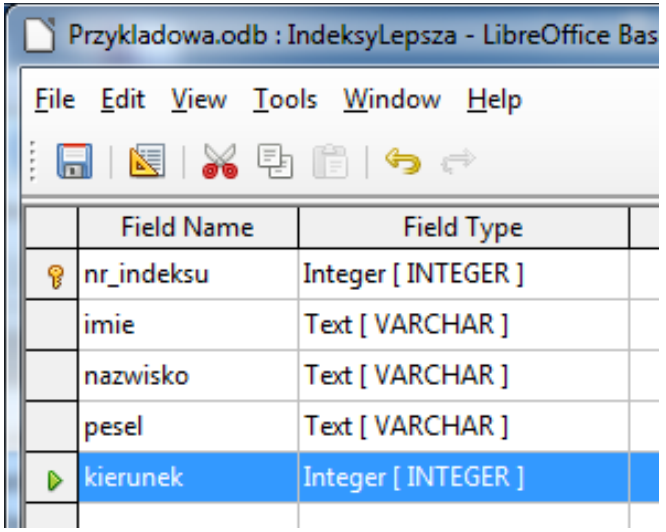
« poprzednie | 1 | 2 | 3 | 4 | 5 | ... | 166 | następne »

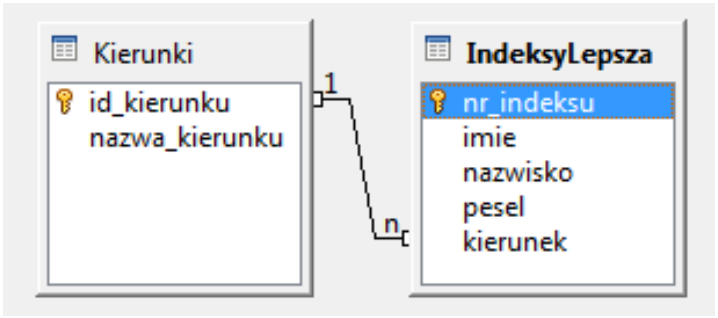
Data Integrity Constraints











Data Integrity Constraints

IndeksyLepsza - Przykladowa - LibreOffice Base: Widok danych tabeli

File Edit View Insert Tools Window Help

nr_indeksu	imie	nazwisko	pesel	kierunek
0	Adam	Nowak	4324	2
1	Ewa	Nowaczyk	4343	1
<AutoField>	Anna	Xyz	322	3
<AutoField>				

LibreOffice Base

Error inserting the new record

Integrity constraint violation - no parent SYS_FK_64 table: Kierunki in statement [INSERT INTO "IndeksyLepsza" ("imie","kierunek","nazwisko","pesel") VALUES (?,?,?,?)]

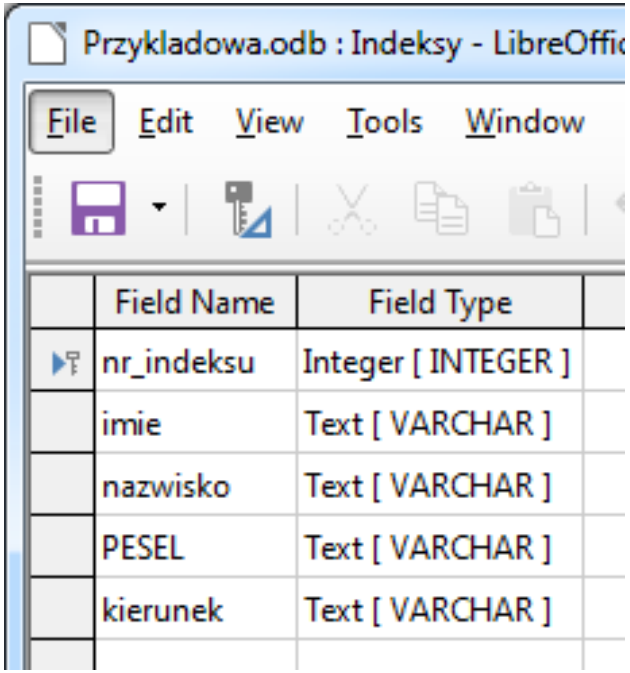
OK More

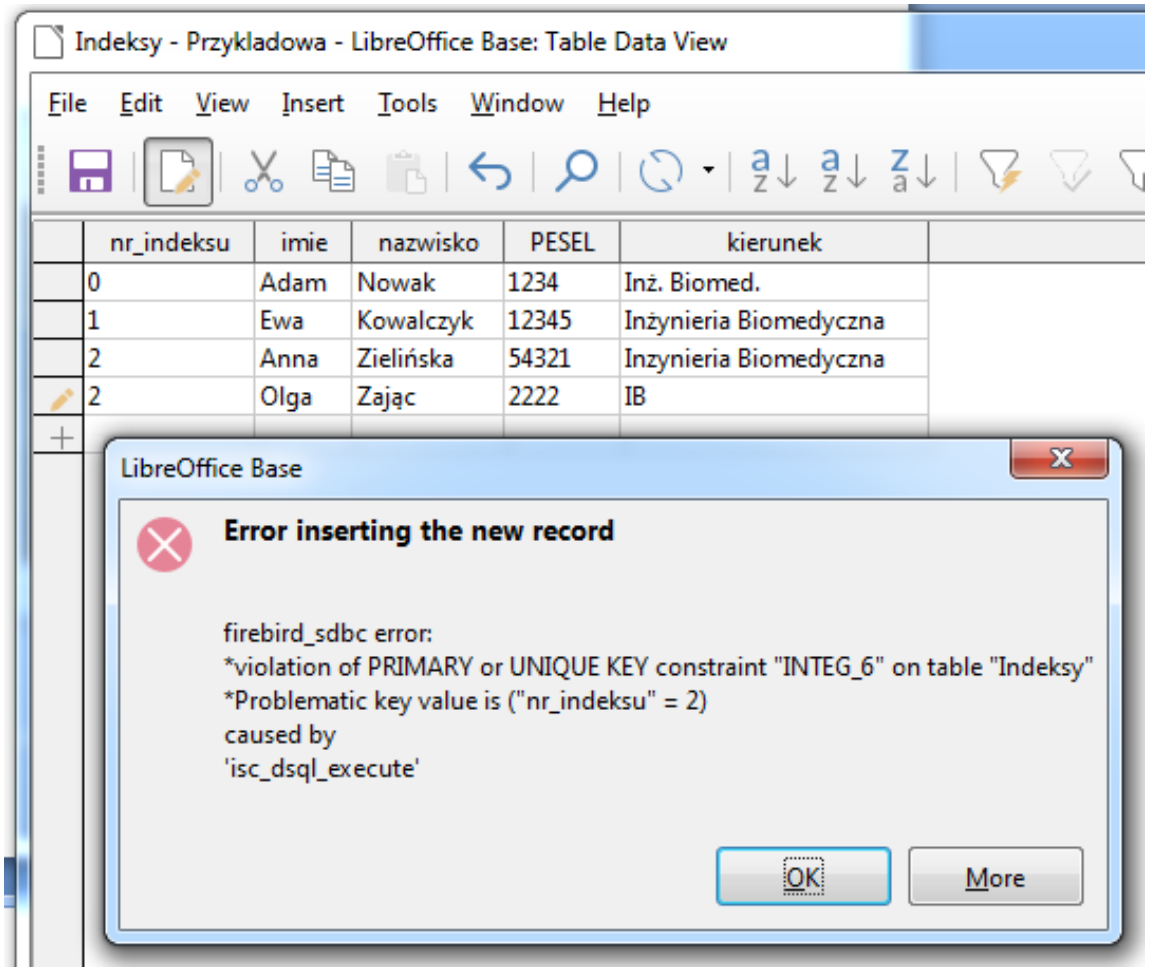
Keys

- Simple key – a key consisting of a single attribute
- Composite key – a key consisting of two or more attributes
- Minimal (candidate) key – the minimal set of attributes that forms a key.
- Superkey – a set of attributes containing a key, but not necessarily a minimal key.
- Subkey – a proper subset of the attributes of a key.
- Primary key – one of the available minimal keys selected by the database designer.
- Foreign key – a set of attributes of one relation that constitutes the primary key of another relation.

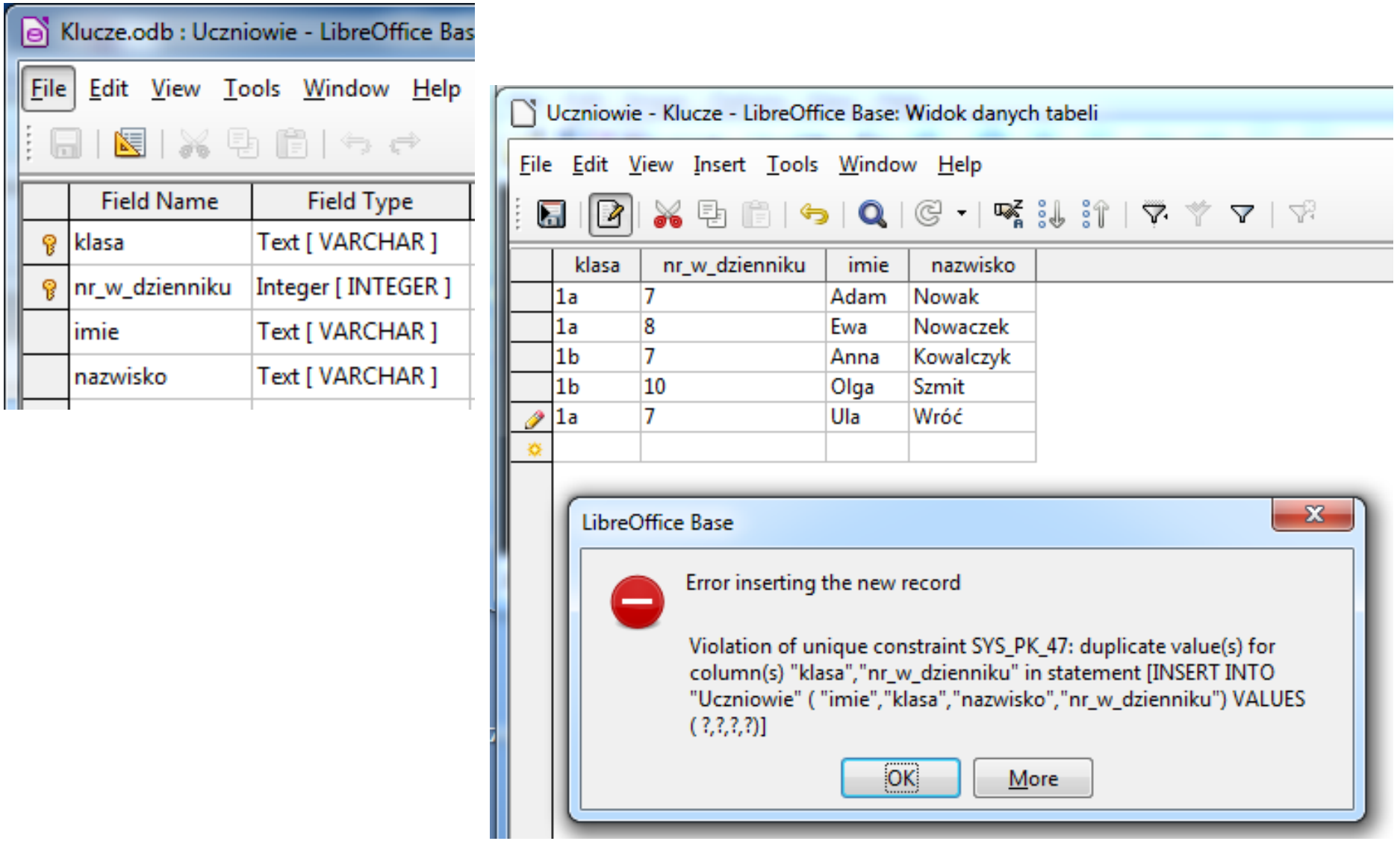
PK – PRIMARY KEY FK – FOREIGN KEY

Keys – Simple Key





Keys – Composite Key



Keys

Which attribute should be selected as the primary key:

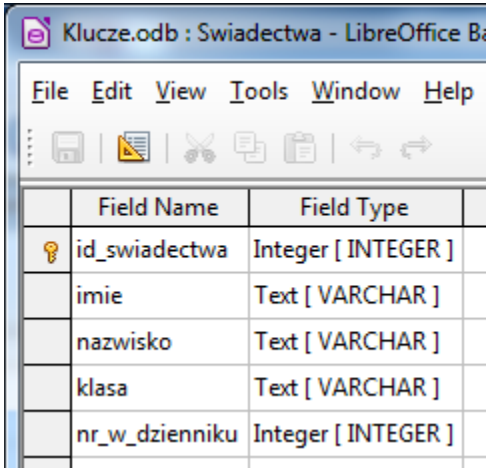
- A unique value, e.g. the PESEL number:
  - if this is a table describing people, PESEL may serve as the key,
  - PESEL may not be unique; for example, in a visitor log, visitors' PESEL numbers may be repeated,
- an integer attribute is compared faster by a DBMS than, for example, a text string; a PESEL stored as a string may also contain invisible spaces that distinguish two values,
- for dictionary values (and generally), each table may have a primary key as an additional attribute (preferably numberic), derived from the name of the relation, e.g.:
  - Powiaty(id_powiatu, name_powiatu)
  - `Studenci(id_studenta, PESEL, student_id, imie, last name)`
- keys in a schema should be highlighted, preferably using underlining; if underscores ‘_’ are present, bold type may improve readability.

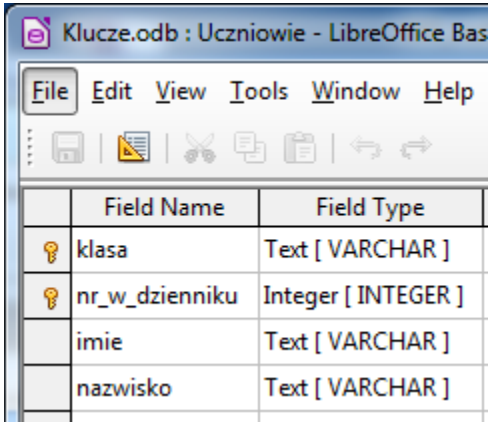
Foreign Keys

- LibreOffice did not work for me here with composite foreign keys
- Such keys are available in MS Access
- Similarities and differences between LibreOffice and MS A

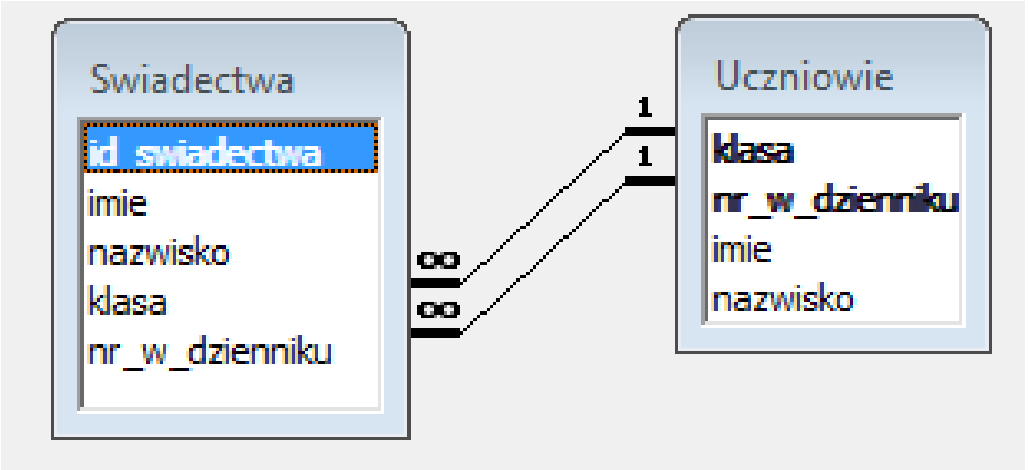




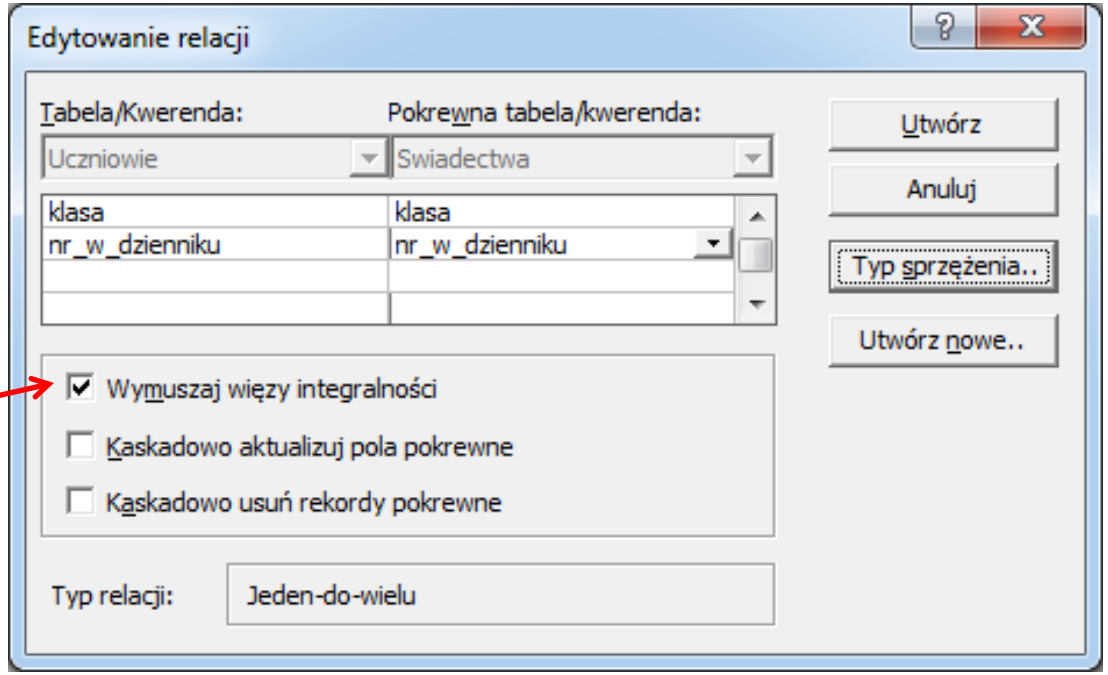




Composite Foreign Keys



MS ACCESS



Tables and Data Types

Logical Model for a Data Warehouse

Dimensions

- tabular data defining the attributes of a given concept,
- dimension tables have primary keys; the key identifies a set of attributes,
- dimensions may form hierarchies, e.g. voivodeship → county → municipality → locality → district,
- data stored in a dimension table usually do not change (new data are more often added),
- a dimension table does not necessarily contain many records.

Facts

- collected data describing a selected phenomenon,
- usually large (or very large) tables,
- have foreign keys referencing dimension tables, whose dimensions contain sets of attributes describing the facts.

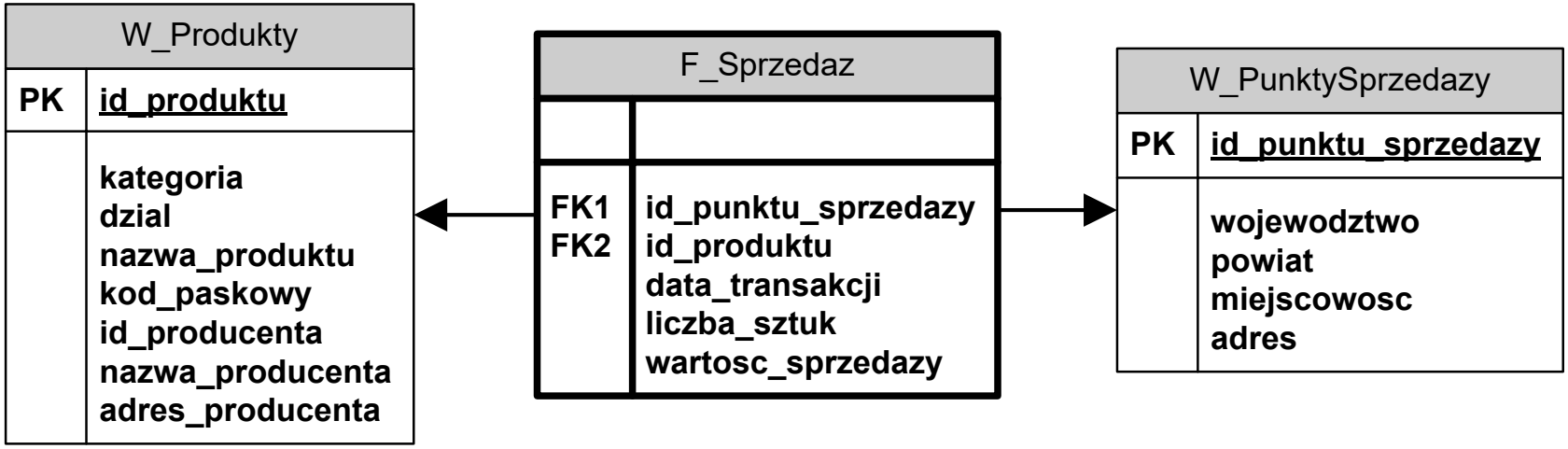
Measures

- numberic values describing a fact, such as price, weight, or length.

Data Schema Models

Star

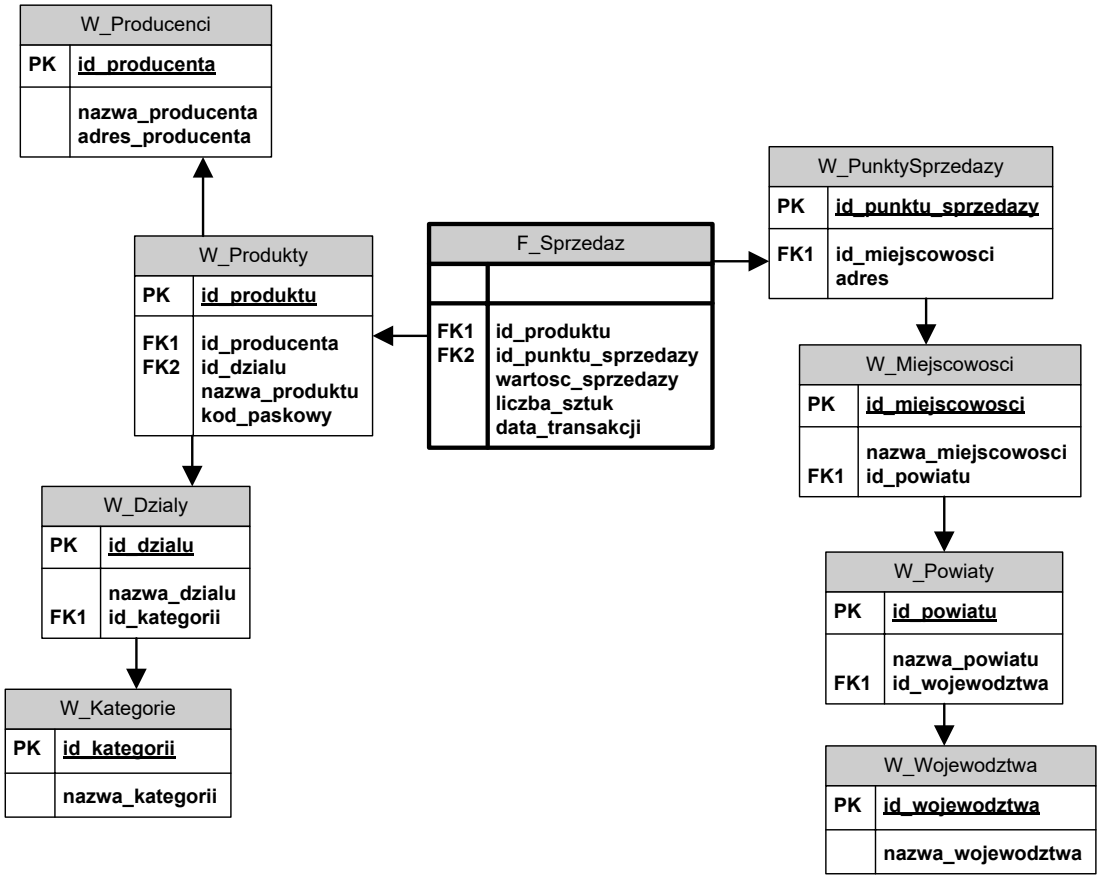
- one normalized fact table,
- many normalized or denormalized dimension tables; each dimension corresponds to only one table (there is no hierarchy),
- simple structure,
- a small number of relationships (→ joins),
- the fact-table key includes all columns except measures.



Data Schema Models

Snowflake

- one fact table (as in the star schema),
- normalized dimension tables (forming a hierarchy – the snowflake),
- the schema is easier to modify
- more relationships – more joins – reduced query efficiency?*

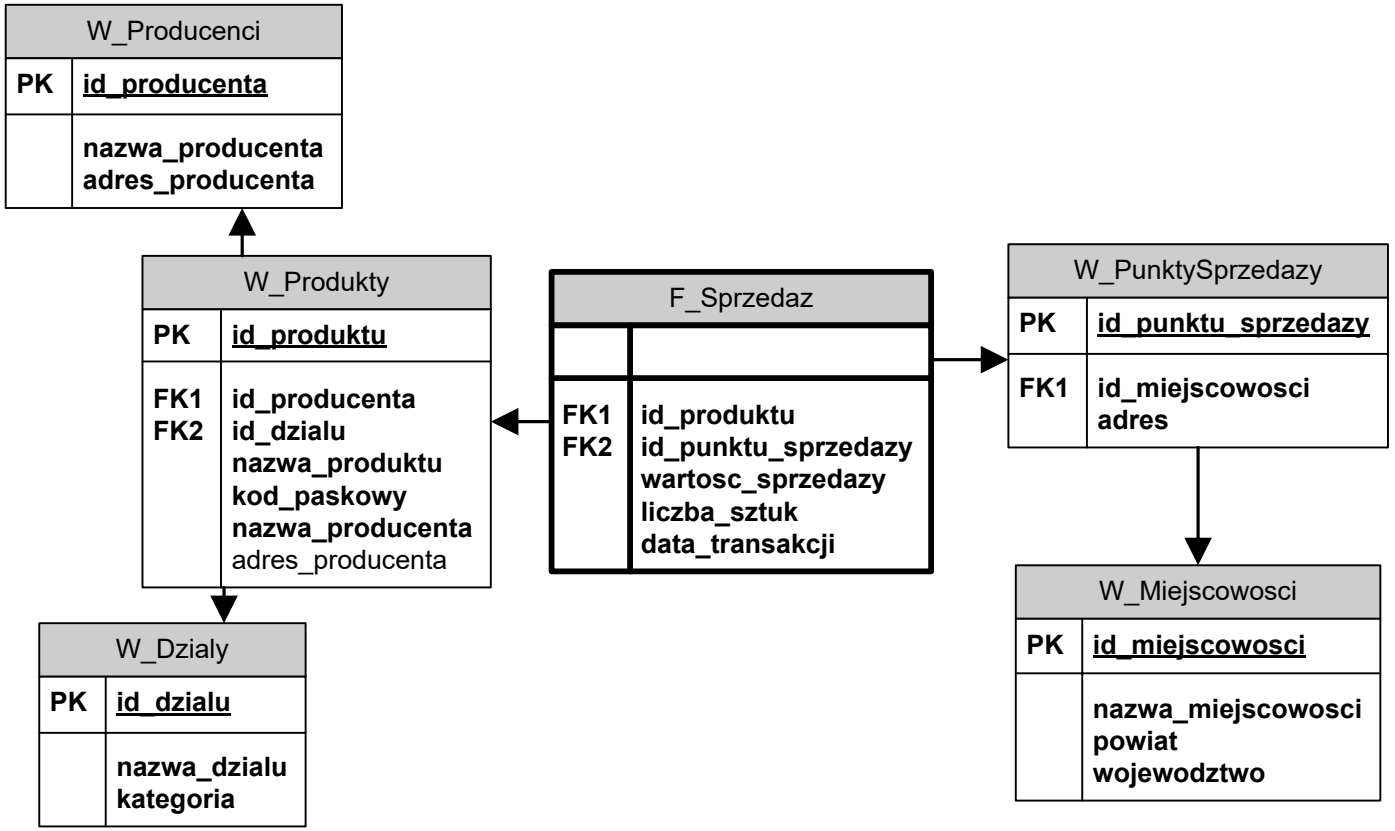


* Jaeśnica Ł., Piórkowski A.: Performance of Joins and Nesting for Normalized and Denormalized Schemas. Studia Informatica. 2010 vol. 31 no. 2A pp. 445–456.

Data Schema Models

Starflake

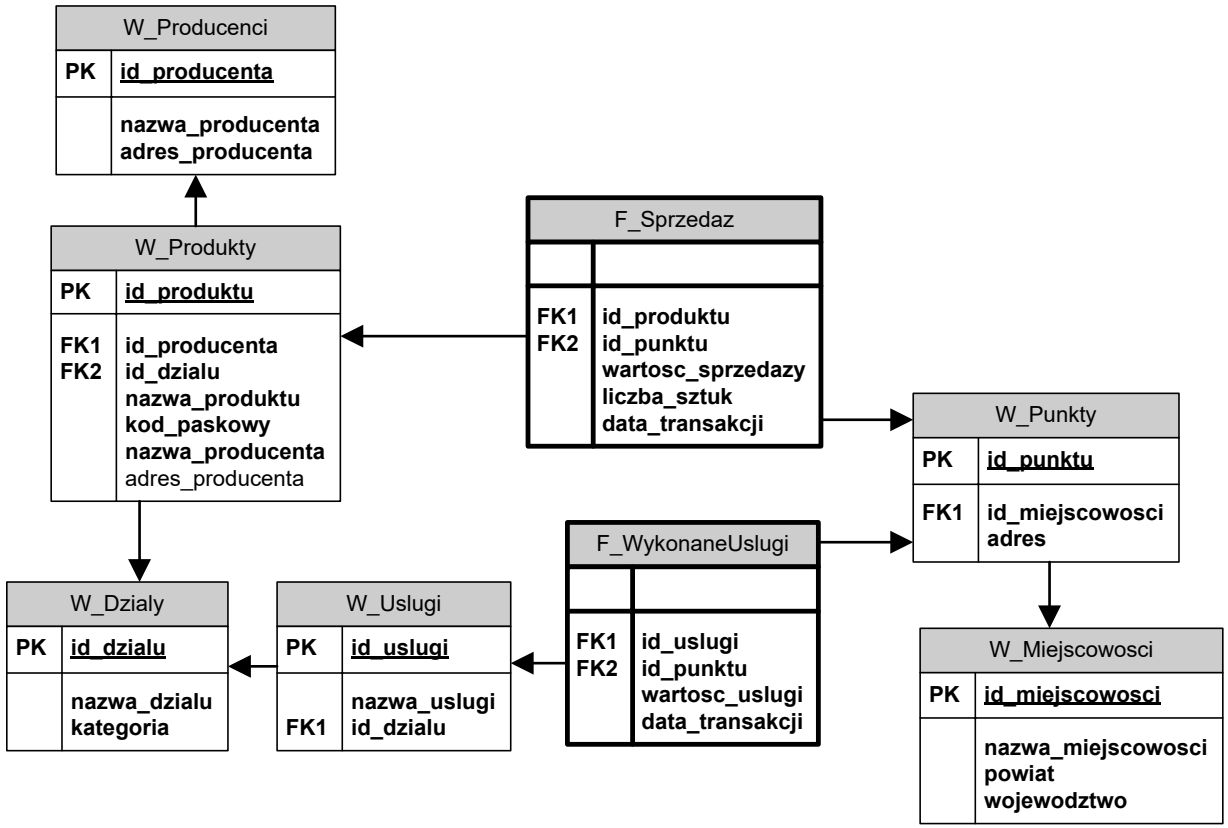
- a schema in which some dimensions are normalized and others are not – a performance optimization in which selected dimension tables will not be modified or extended.



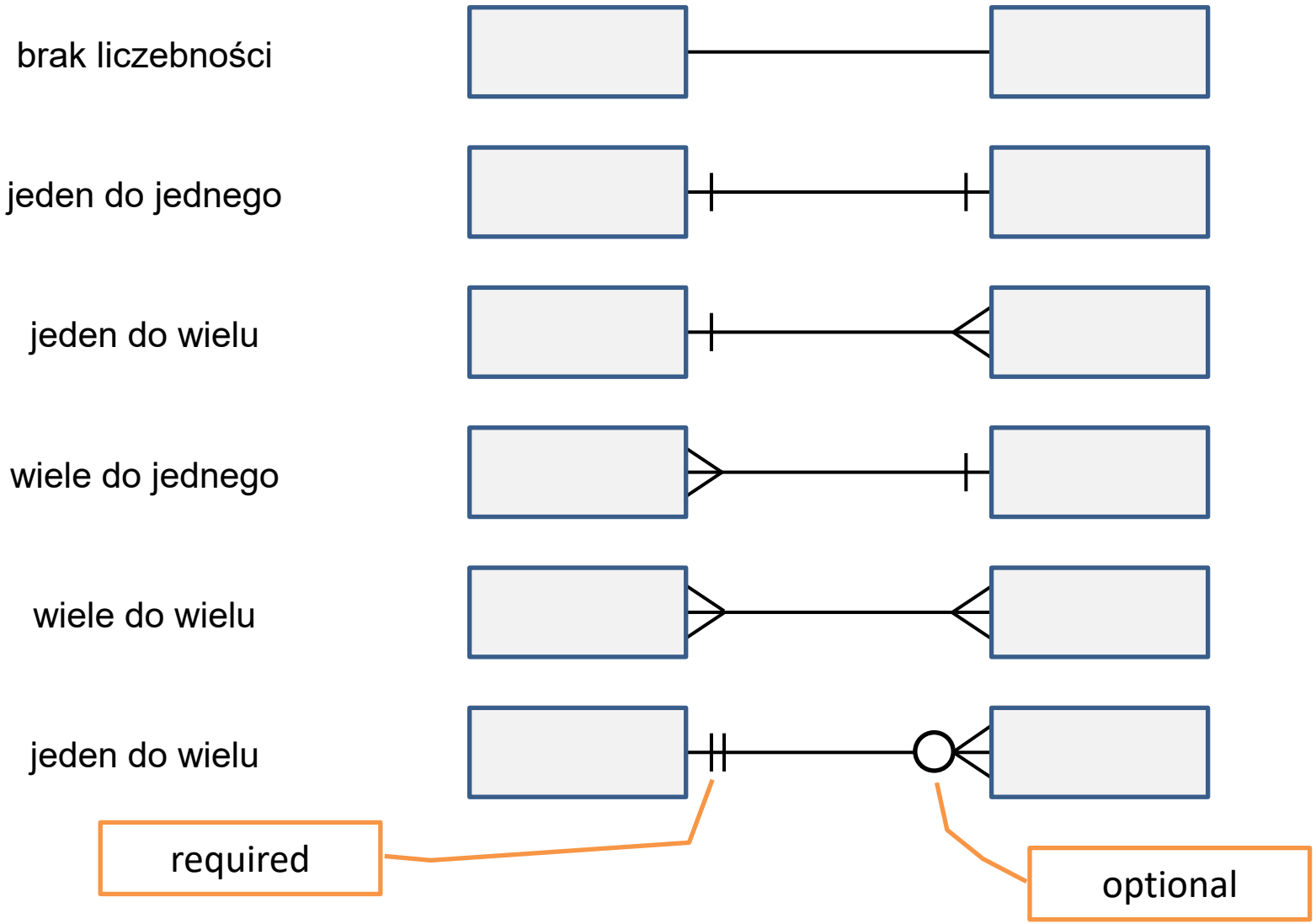
Data Schema Models

Constellation

- there are at least two fact tables, which may be linked,
- fact tables may share dimension tables,
- a flexible solution, but more difficult to implement and develop.

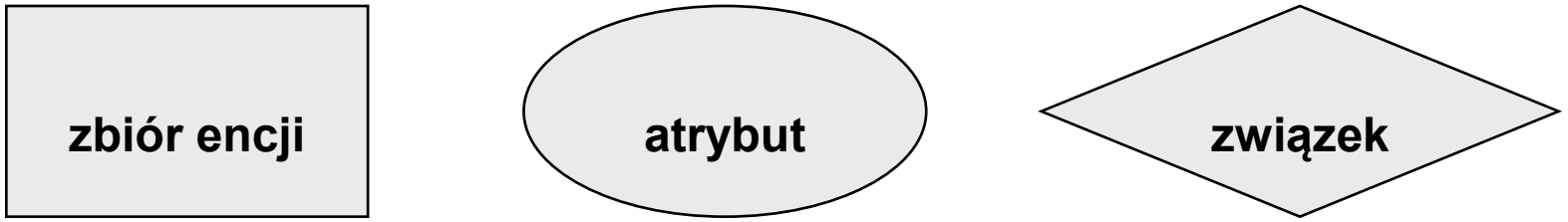


Martin Notation



Entity–Relationship Diagrams (ERD) Chen Notation

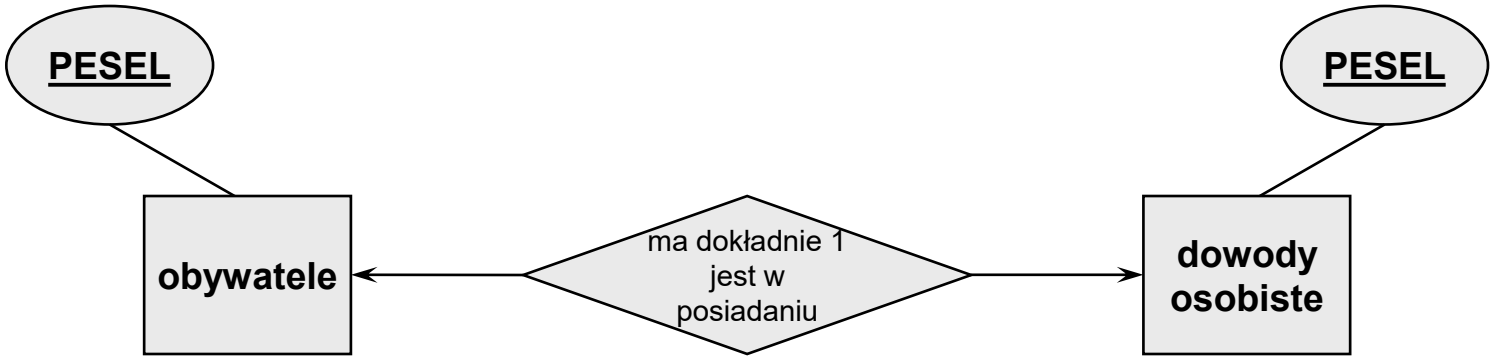
Entity sets – rectangles
Attributes – ovals
Entity relationships – diamonds



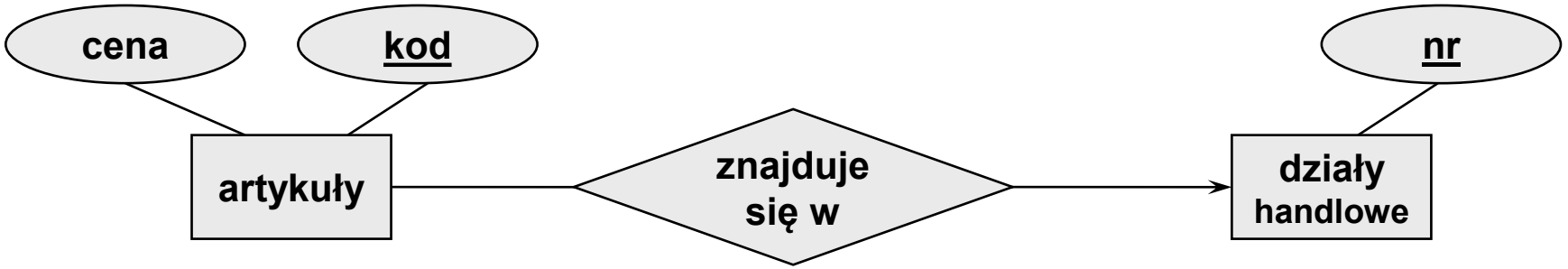
Entity-set names – plural form
The attribute–entity-set connection is undirected
The entity-set–relationship–entity-set connection may be directed or undirected.
An arrow indicates a single entity.

Entity–Relationship Diagrams (ERD) Chen Notation

One-to-one relationship

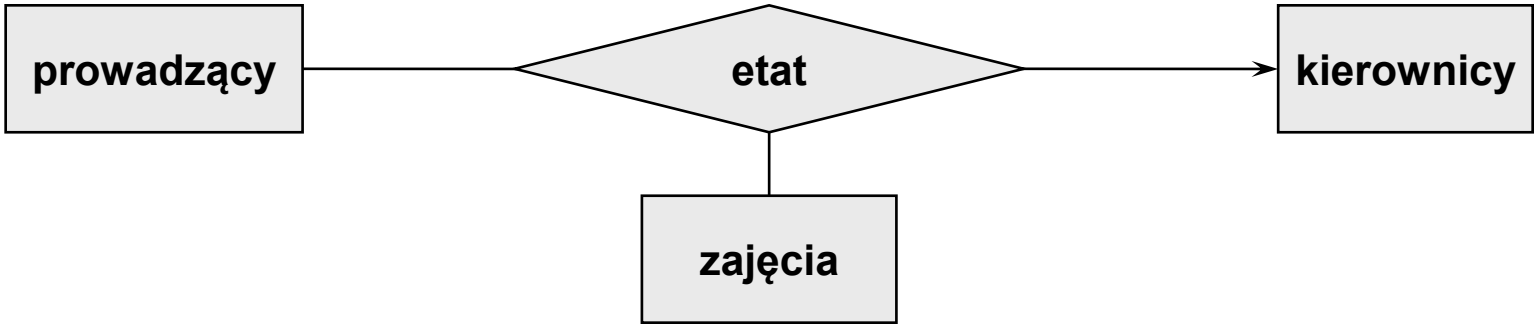


Many-to-one relationship

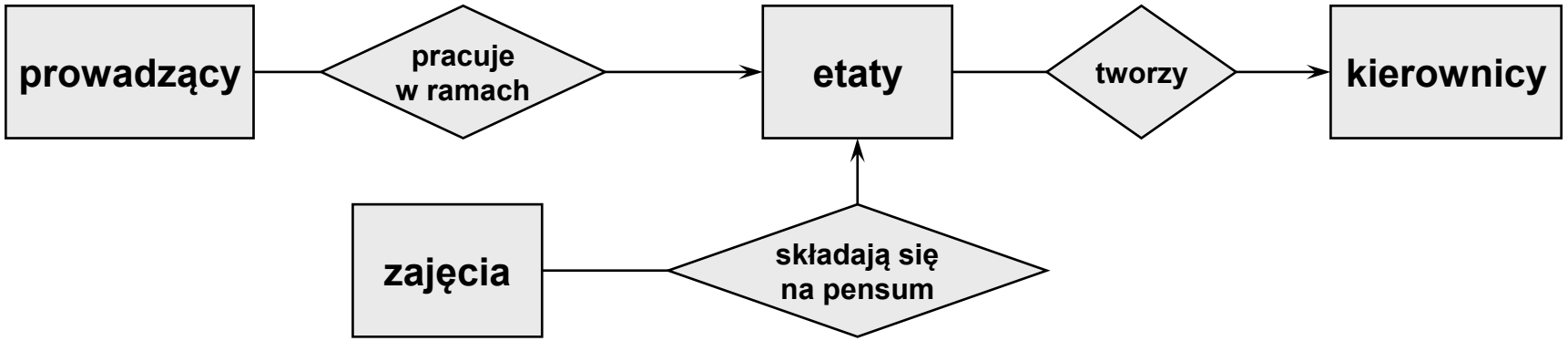


Entity–Relationship Diagrams (ERD) Chen Notation

N-ary relationships

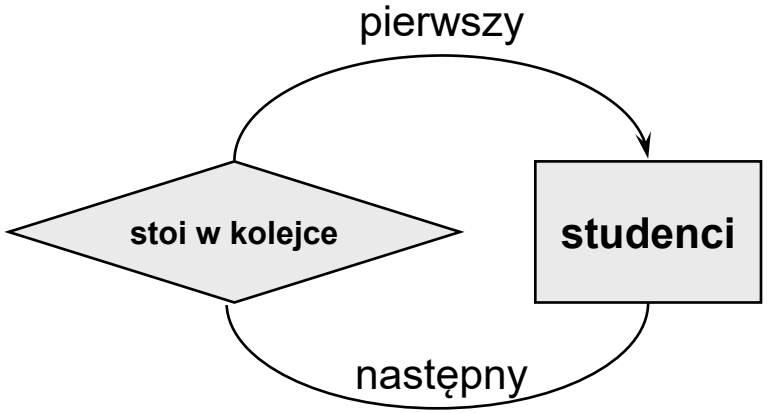


Decomposition of an n-ary relationship into binary relationships

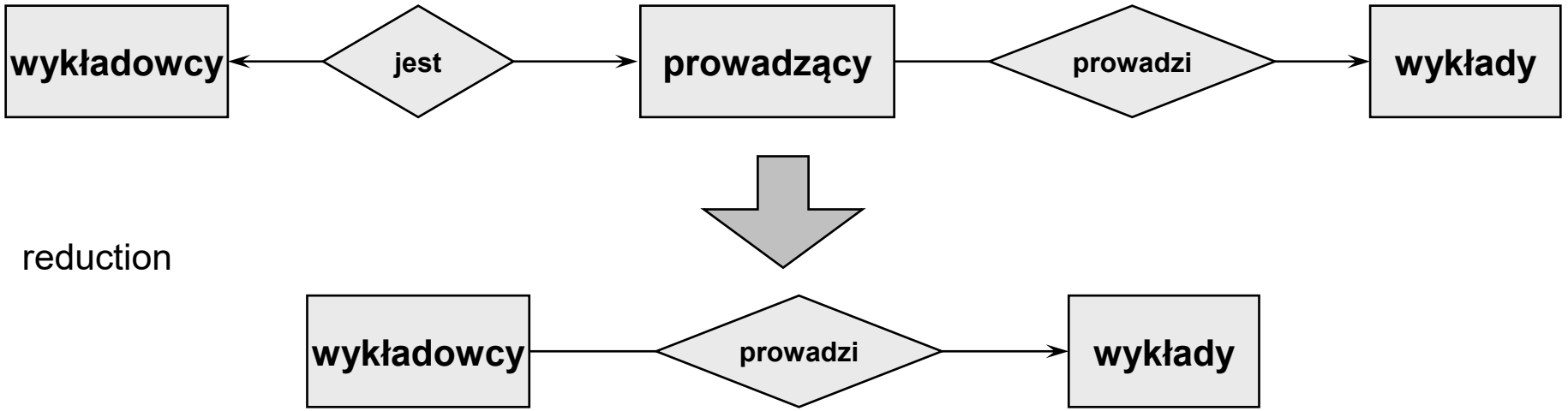


Entity–Relationship Diagrams (ERD) Chen Notation

Multiple use of an entity set



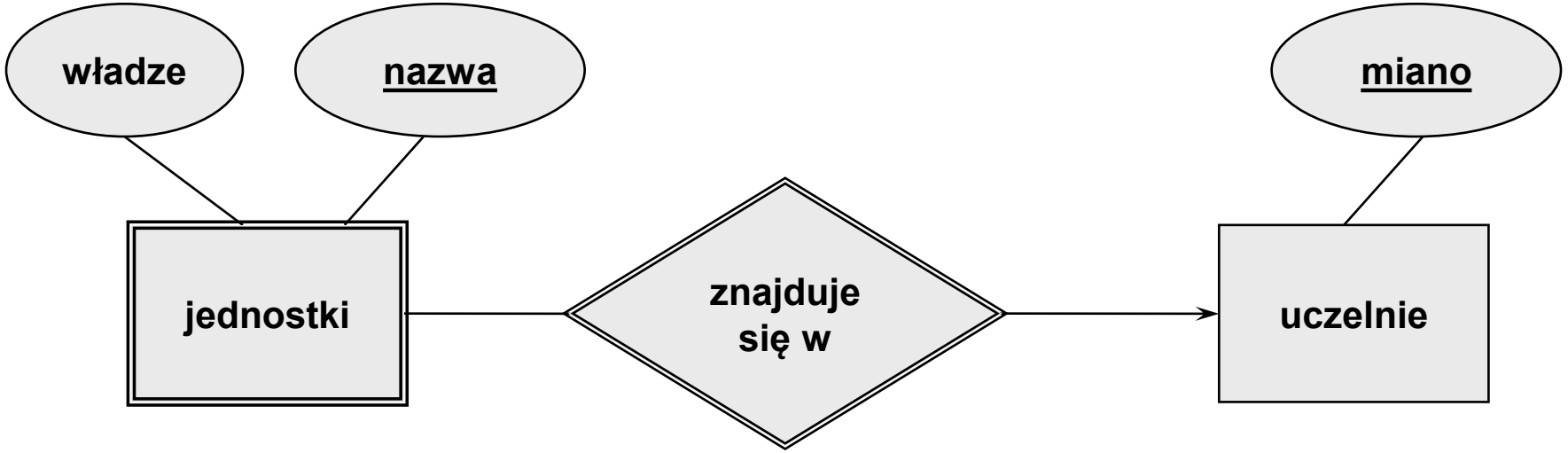
Redundancy



WEAK ENTITY SETS

Weak entity:
may exist only in association with other entities or may not have its own key.

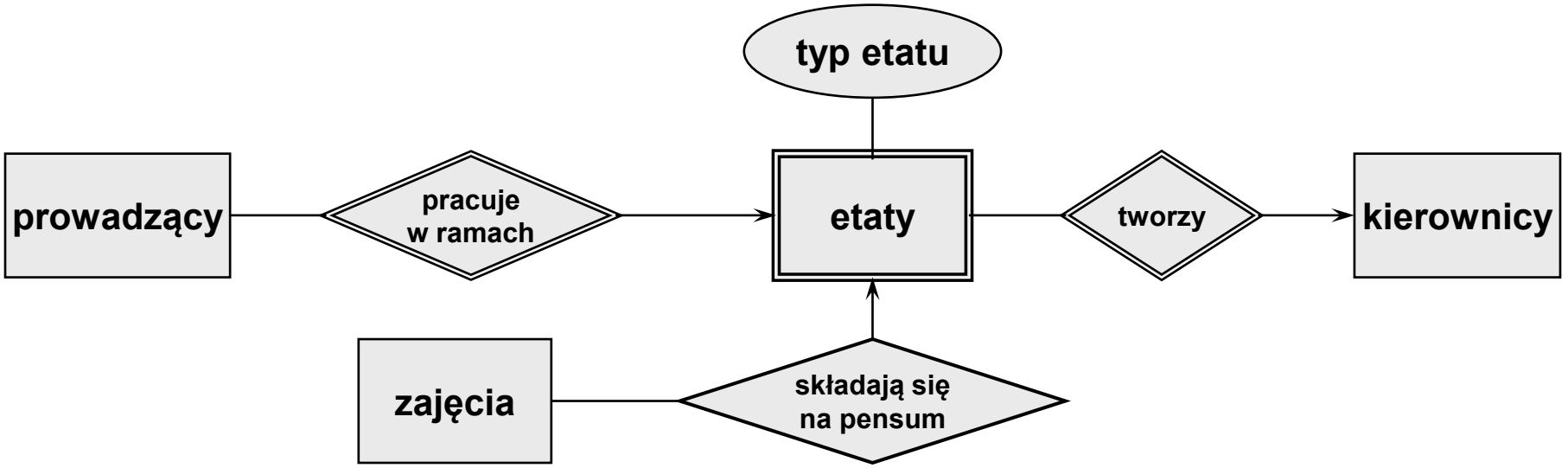
Case 1 – entity set A is a subset of entity set B
unit: authorities (e.g. {dean, head, ...}), name (e.g. {library, computer science faculty, ...})
university: name (e.g. {AGH, UE, UP, UJ, ...})



? key {e.g. position: library head – at UP, at UE}

WEAK ENTITY SETS

Case 2 – no own key



Jan Nowak works 210 hours under an assistant professor position created by the head of the unit, Marcin Kowalski.
Jan Nowak works 210 hours under an assistant professor position created by the head of another unit, Jerzy Wójcik.
Justyna Krawczyk works 60 hours under an assistant professor position created by the head of the unit, Jerzy Wójcik.
Alicja Kowalska works 120 hours under an assistant position created by the head of the unit, Jerzy Wójcik.

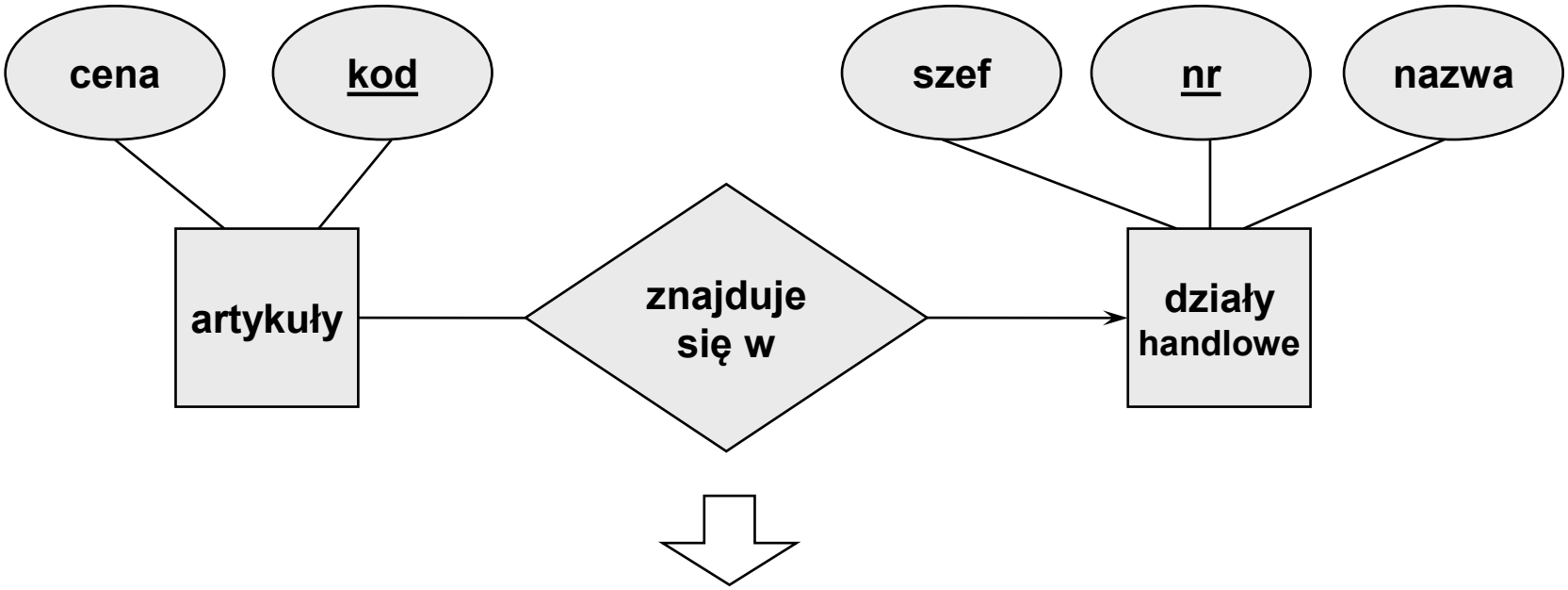
Transforming Entity–Relationship Diagrams into the Relational Data Model

For each entity set, a table is created with the same name as the entity set and the same set of attributes.

For each entity relationship, a table is created whose attributes include the keys of the related entity sets and, where applicable, relationship keys.

Depending on relationship cardinality:
- 1:1 – the new table takes the key from one of the entity sets,
- N:1 – the new table takes the key from the entity set on the N side of the relationship,_x000B_- M:N – the new table takes the keys of both entity sets.

Transforming Entity–Relationship Diagrams into the Relational Data Model



artykuły

kod	cena
123	100,50
124	25,00

położenie

kod	nr
123	1
124	2

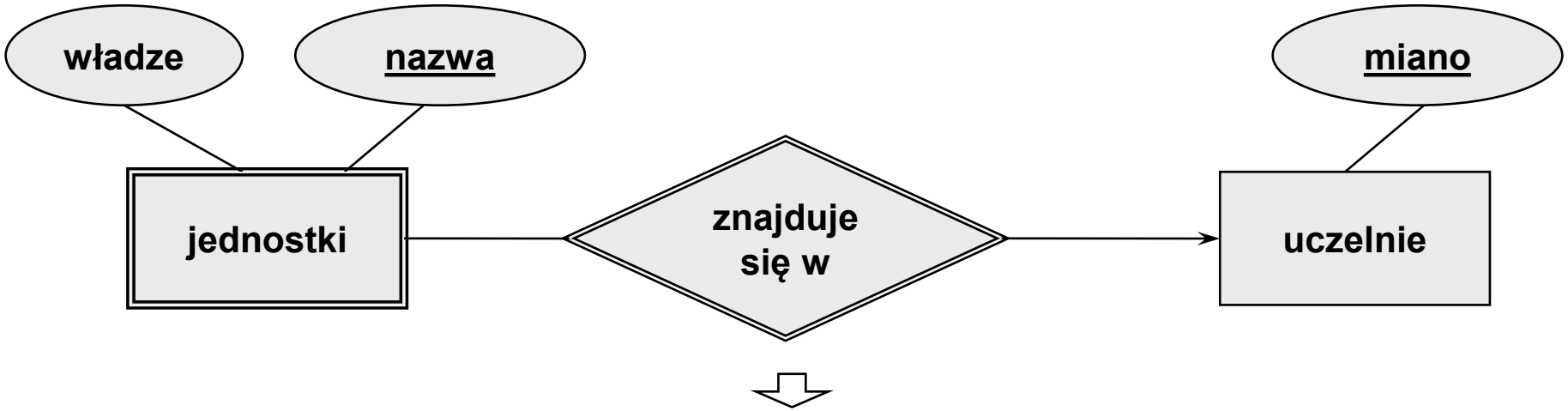
działy handlowe

nr	nazwa	szef
1	chemia	J. Nowak
2	prasa	L. Kot

Transforming Entity–Relationship Diagrams into the Relational Data Model

Procedure for weak relationships:

- the relation corresponding to a weak entity set must also contain attributes of the keys of the related relations
- relationships involving a weak entity set must use the keys of the related entity sets
- it is not always necessary to decompose relationships of weak entity sets using additional relations



jednostki

nazwa	władze	miano
Wydz Inf. St.	dziekan	AGH
Wydz Inf. St.	dziekan	UJ
Wydz Mat. St.	dziekan	AGH

uczelnie

miano
AGH
UJ

ANOMALIES IN RELATIONAL DATABASE SCHEMAS

Creating relations with a large number of attributes (e.g. by combining other relations) leads to the following anomalies:

- **redundancy – excess data in a table leads to excessive memory consumption and slows system operation,**
- **insertion anomaly – inability to insert a tuple without complete data,**
- **update anomaly – failure to update some tuples,**
- **deletion anomaly – deleting an attribute value from one tuple causes the data of the entire tuple to be lost.**

marka	model	kolor	karos.	linia	typ_silnika	pojemność	wtrysk	paliwo
VW	golf	czarny	ocynk.	3drz.	DCV200D	1300	SI	ON
VW	golf	czarny	ocynk.	3drz.	DCV200	1300	SI	U95
VW	golf	czarny	ocynk.	5drz.	DCV200D	1300	SI	ON
VW	golf	czarny	ocynk.	5drz.	DCV200	1300	SI	U95
VW	golf	zielony	ocynk.	3drz.	DCV200D	1300	SI	ON
VW	golf	zielony	ocynk.	3drz.	DCV200	1300	SI	U95

RELATION DECOMPOSITION

Relation A: {A1,...,Ax} is decomposed into two relations B: {B1,...By}, i C: {C1,...,Cz}, such that:_x000B_

1. {B1,...By} □ {C1,...,Cz} = {A1,...,Ax}_x000B_

2. the tuples of relations B and C are obtained by projecting all tuples of relation A, with duplicates removed

marka	model	kolor	karos.	linia	typ_silnika
VW	golf	czarny	ocynk.	3drz.	DCV200D
VW	golf	czarny	ocynk.	3drz.	DCV200
VW	golf	czarny	ocynk.	5drz.	DCV200D
VW	golf	czarny	ocynk.	5drz.	DCV200
VW	golf	zielony	ocynk.	3drz.	DCV200D
VW	golf	zielony	ocynk.	3drz.	DCV200

typ_silnika	pojemność	wtrysk	paliwo
DCV200D	1300	SI	ON
DCV200	1300	SI	U95

FUNCTIONAL DEPENDENCIES

Functional dependency: A1, ... , Ax → B

if two tuples of a relation agree on attributes A1,...,Ax, they must also agree on attribute B.

For a functional dependency involving multiple attributes, the notation is as follows:
A1... Ax → B1... By

The functional dependency A1... Ax → B1... By is:
- trivial – if B1...By is contained in A1...Ax, e.g.: first name, last name, PESEL → last name
- non-trivial – if at least one attribute in B is not contained in A, e.g.: first name, last name, PESEL → PESEL, salary
- completely non-trivial – no attribute from B belongs to A, e.g.: first name, last name, PESEL → salary

Full functional dependency – a key uniquely determines certain non-key values.

Partial functional dependency – a subkey uniquely determines certain non-key values.

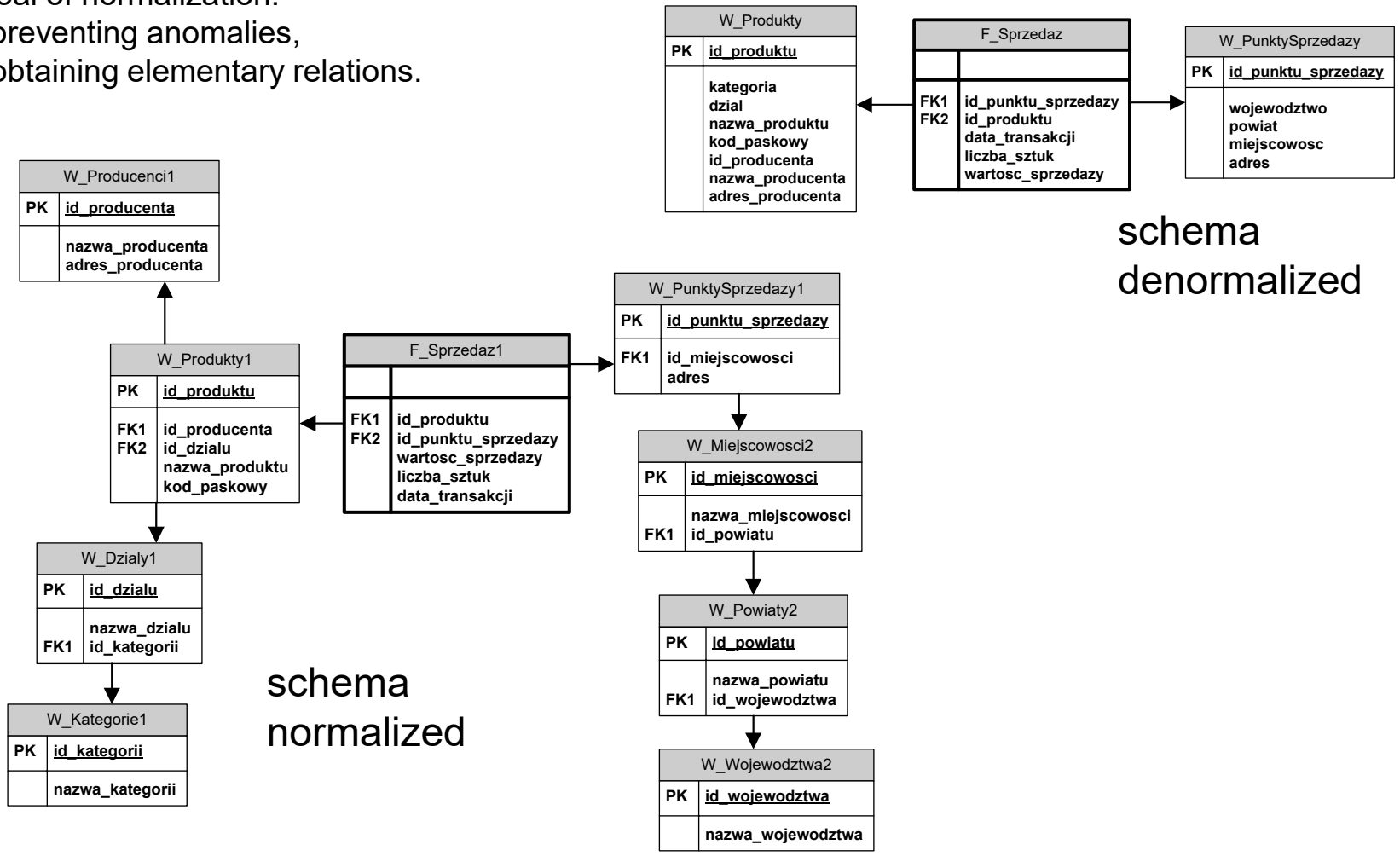
Multivalued Dependencies

Multivalued dependency: A →> B

Knowing the value of attribute A allows us to determine the set of possible values of attribute B.

RELATIONAL DATABASE SCHEMA NORMALIZATION

Goal of normalization:
- preventing anomalies,
- obtaining elementary relations.



FIRST NORMAL FORM (1NF)

A relation is in first normal form (1NF) if attribute values are single atomic values.

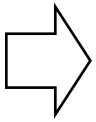
That is, the value of an attribute in a single tuple is an atomic value of a defined type; data aggregates (lists, tables, etc.) do not occur.

Note! In object-relational data, an object class may be defined and may become an attribute.

klasa	sala
1a	BUDYNEK
1b	315

adres	liczba miejsc
Krótka 2a	40
Długa 3b	35

klasa	sala
1a	314, 316
1b	315



klasa	sala1	sala2
1a	314	316
1b	315	

SECOND NORMAL FORM (2NF)

Full functional dependency – a key uniquely determines certain non-key values.
Partial functional dependency – a subkey uniquely determines certain non-key values.

Relation X is in 2NF if it is in 1NF and contains no partial functional dependencies (the key determines all non-key attributes).

Functional dependencies between non-key attributes are permitted.
- Transitive dependencies may occur.
- Non-trivial functional dependencies whose left-hand side is a subkey are not permitted.

imię	**nazwisko**	**imieniny***	**urodziny**	**znak zodiaku**
Adam	Nowak	24.12	8.08	lew
Agnieszka	Kowalska	21.01	2.06	bliźnięta

Notes(first name, last name, name day, date of birth, zodiac sign)
date of birth → zodiac sign – {date of birth} is not part of the key, therefore the dependency does not exclude 2NF.
first name → name day – {first name} is a subkey, therefore the dependency excludes 2NF.

* We assume that a given first name has one name day per year (e.g. Adam); otherwise, this would be an example of a

THIRD NORMAL FORM (3NF)

Relation X is in 3NF <=>

if, for every non-trivial dependency A1...Ax → B,

then A1...Ax is a superkey or B is a subkey.

Notes:

•Functional dependencies between non-key attributes are not possible.

For example: Notes(first name, last name, name day, date of birth, zodiac sign)
date of birth → zodiac sign – {date of birth} is not part of the key, so the 2NF condition is satisfied, but 3NF is not.

•B may be part of another key (a relation may contain different sets of attributes forming keys).

example 0: PESEL / ID card number / passport number

miasto	ulica	kod
Kraków	Mickiewicza	30-059
Kraków	Tokarskiego.	30-065
Gdańsk	Mickiewicza	80-405

Addresses(city, street, code)
Keys: { city, street }, { street, code } (for simplicity, we assume that a given code applies along the entire street)

Functional dependencies: { city, street } → code,

code → city

(code is a subkey, so the 2NF condition is satisfied, but BCNF is not)

BOYCE–CODD NORMAL FORM (BCNF)

Relation X is in BCNF <=>
if, for every non-trivial dependency A1... Ax → B,
then A1...Ax is a superkey.

Notes:

- There are no functional dependencies:
  - transitive,
  - between non-key attributes,
  - between non-key attributes and a subkey,
- There are no additional candidate keys.

- The only functional dependency is associated with the primary key.

BCNF is a very strong constraint and can sometimes be difficult to implement; 3NF is often assumed to be sufficient, for example, when designing data warehouse schemas.

NORMALIZATION – EXAMPLE

students (nr_dz, first name, last name, class, room)

nr_dz	imię	nazwisko	klasa	sala
1	Andrzej	Nowak	1a	314
2	Gustaw	Kot	1a	314.
3	Ala	Wójcik	1a	314.
1	Ola	Nowak	1b	315.
2	Gustaw	Werner	1b	315.
3	Ala	Wójcik	1b	315.

Key K = {nr_dz, class}
Dependencies:
nr_dz, class → last name { nr_dz, class } – superkey K
nr_dz, class, last name → room { nr_dz, class, last name } – superkey K
class → room { class } - is a subkey K, (a nie superkeyem)
Therefore, the presented database model does not satisfy the requirements of 2NF, 3NF, or BCNF.

Assumption – e.g. primary education: a class is taught in only one room, and only one class is taught in that room.

DECOMPOSITION INTO BCNF

Decomposition is performed by identifying a dependency A1...Ax → B1...By that violates the BCNF condition. The set B1...By is then extended with additional attributes functionally dependent on A1...Ax, and a new relation is created containing A1...Ax and B1...By, while B1...By are removed from the decomposed relation.

nr_dz	imię	nazwisko	klasa
1	Andrzej	Nowak	1a
2	Gustaw	Kot	1a
3	Ala	Wójcik	1a
1	Ola	Nowak	1b
2	Gustaw	Werner	1b
3	Ala	Wójcik	1b

klasa	sala
1a	314
1b	315

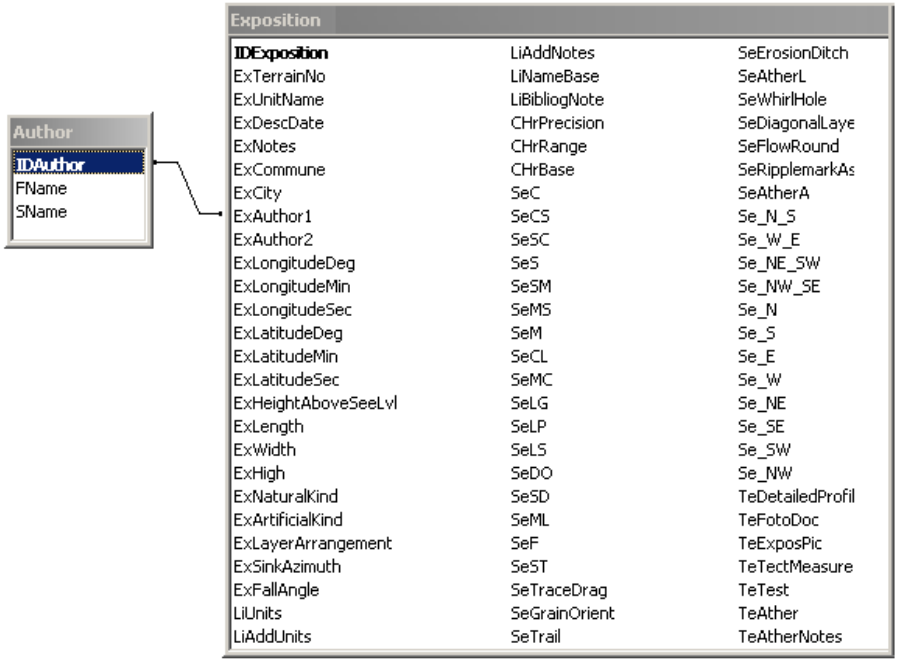
For the given example:

student (student_no, first name, last name, class)

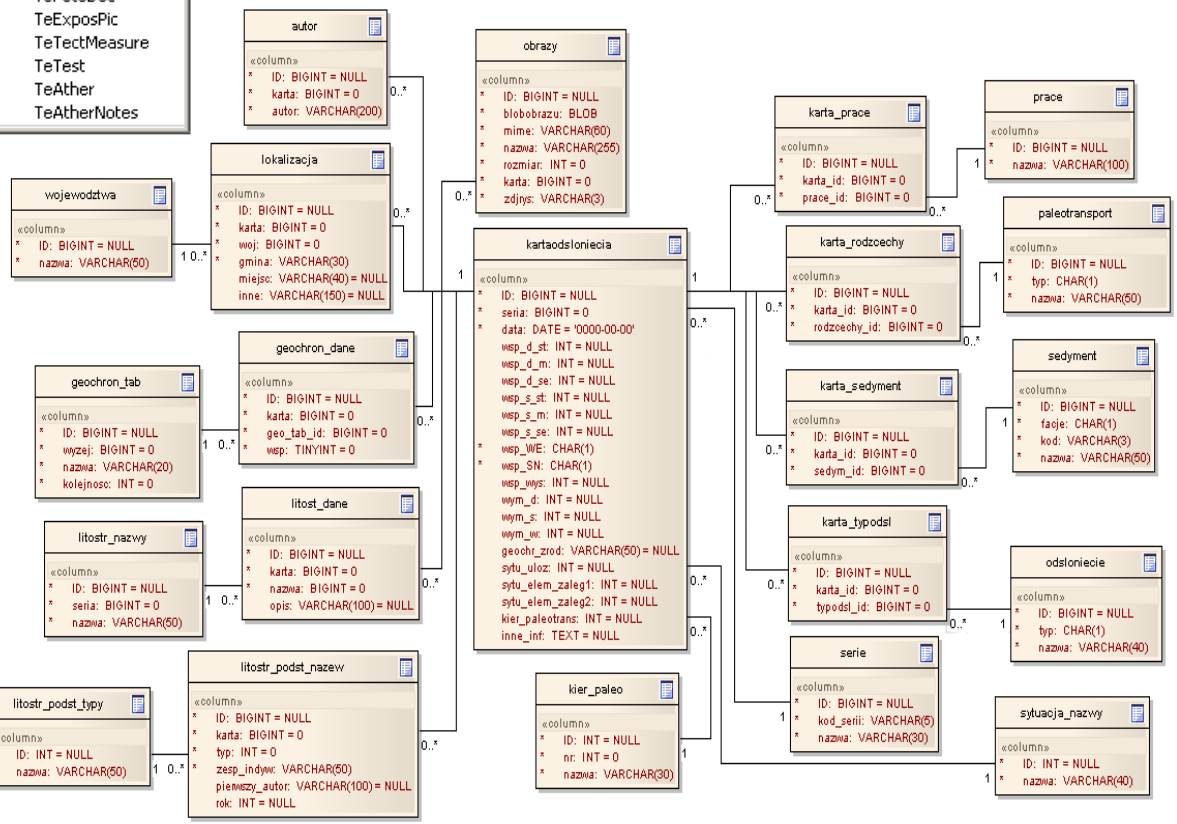
class (class, room)

Data separation after decomposition into BCNF

During decomposition into BCNF (based on functional dependencies), data from the



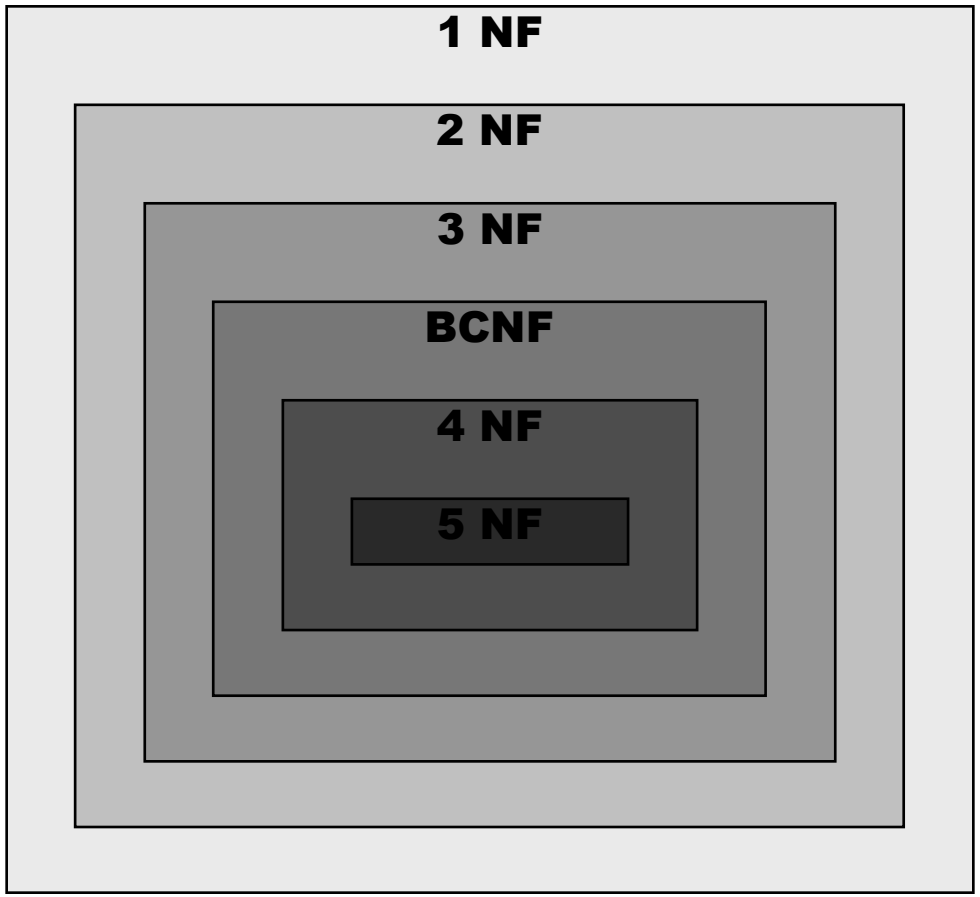
denormalized



normalized

OTHER NORMAL FORMS

- fourth normal form (4NF), (at most one multivalued dependency)
- fifth normal form (5NF, join-projection normal form),
- domain-key normal form.



DATA OPERATIONS

Basic set-theoretic operations
-union
-difference
-intersection

Compound operations

Data selection
- projection
- selection

Table joins

- Cartesian product (CROSS JOIN)
- theta join
- natural join (NATURAL JOIN)
- equijoin / inner join (INNER JOIN)
- outer joins (left, right, full – OUTER JOIN)
- self-join (auto-join)
- semi-join
- division

DATA OPERATIONS

UNION

Relations A and B are given:
A(atr1, atr2), B(atr1, atr2)

A

ATR1	ATR2
1	x
2	y
3	z

.

B

ATR1	ATR2
1	u
2	y

Union

The union of relations A and B (A ∪ B) is the set of all tuples belonging to relation A or B, including the
duplicates are removed

-- or not

A ∪ B

ATR1	ATR2
1	x
2	y
3	z
1	u
2	y

A ∪ B

ATR1	ATR2
1	x
2	y
3	z
1	u

.

DATA OPERATIONS

DIFFERENCE

Relations A and B are given:
A(atr1, atr2), B(atr1, atr2)

A

ATR1	ATR2
1	x
2	y
3	z

B

ATR1	ATR2
1	u
2	y

.

Difference

The difference of relations A and B (A – B) is the set of all tuples belonging to relation A and not belon

A - B

atr1	atr2
1	x
3	z

DATA OPERATIONS

INTERSECTION

Relations A and B are given:
A(atr1, atr2), B(atr1, atr2)

A

ATR1	ATR2
1	x
2	y
3	z

B

ATR1	ATR2
1	u
2	y

.

INTERSECTION

The intersection of relations A and B (A ∩ B) is the set of all tuples belonging to both relation A ar
The intersection can be expressed using difference: A ∩ B = A – (A – B)

A ∩ B

ATR1	ATR2
2	y

DATA OPERATIONS

PROJEKCJA

Projection

Projection Π of relation A transfers the data of relation A to another (new) relation B, whose set of attributes {b1, ... , bk} is a subset of the attributes of A {a1, ... , an }

A

atr1	atr2	atr3	atr4
1	x	aa	dx
2	y	bb	dy

$$\Pi_{atr1,\ atr2\ ,atr4}\ (A)$$

atr1	atr2	atr4
1	x	dx
2	y	dy

the number of tuples does not change, but duplicates may arise

DATA OPERATIONS

SELECTION

Selection

Selection operation σ on relation A creates a subset of tuples satisfying criterion F, which operates on the attributes of A.

A

atr1	atr2	atr3	atr4
1	x	aa	123
2	y	bb	234
3	x	aa	576
4	y	bb	121
3	x	aa	260

$$\sigma_{atr1 < 4 \,,\, atr4 < 250} (A)$$

atr1	atr2	atr3	atr4
1	x	aa	123
2	y	bb	234

DATA OPERATIONS

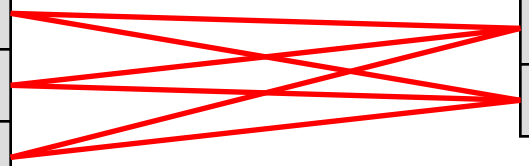
Cartesian Product

A

atr1	atr2	atr3
11	x	aa
22	y	bb
33	x	bb

B

atr2	atr3	atr4	atr5
x	aa	123	11
y	bb	234	22



A►◄B

atr1	A.atr2	A.atr3	B.atr2	B.atr3	atr4	atr5
11	x	aa	x	aa	123	11
11	x	aa	y	bb	234	22
22	y	bb	x	aa	123	11
22	y	bb	y	bb	234	22
33	x	bb	x	aa	123	11
33	x	bb	y	bb	234	22

DATA OPERATIONS

θ join (theta join)

A theta join of two relations A and B (A►◄ θ B) is obtained by selecting the results of A×B using a condition θ. The condition usually concerns selected attributes of relations A and B.

A

atr1	atr2	atr3
11	x	aa
22	y	bb
33	x	bb
44	y	bb

B

atr2	atr3	atr4	atr5
x	aa	123	11
y	bb	234	22

θ: atr1 > atr5

A►◄ $_{\vartheta}$ B

atr1	A.atr2	A.atr3	B.atr2	B.atr3	atr4	atr5
22	y	bb	x	aa	123	11
33	x	bb	x	aa	123	11
44	y	bb	x	aa	123	11
33	x	bb	y	bb	234	22
44	y	bb	y	bb	234	22

DATA OPERATIONS

DIVISION

Relations A={a1,a2,...,am} and B={b1,b2,..bk} are given. B ≠ Ø, m>k

The division of relations A and B (A÷B) is a new relation C containing those tuples of A that contain all combinations of tuples of B.
The set of attributes of relation C contains only attributes belonging to A and not to B.

A

atr1	atr2	atr3	atr4
1	x	aa	12
2	y	aa	12
3	z	aa	12
1	x	bb	34
2	y	bb	34
3	z	bb	34
1	x	cc	56
2	y	cc	56

B

atr3	atr4
aa	12
bb	34
cc	56

A÷B

atr1	atr2
1	x
2	y

3	z

3	z	cc	56

DATA OPERATIONS

NATURAL JOIN

A (atr1, atr2, atr3,....), B(atr2, atr3, atrbis1, atrbis2,...)

Natural join of relations A and B (A►◄B) creates a new relation whose set of attributes is the set-theoretic union of the attribute sets of A and B. The operation proceeds as follows:
1. calculate A × B,
2. from the Cartesian product, select only those tuples that have the same values for attributes common to A and B (A.atr2 ↔ B.atr2, A.atr3 ↔ B.atr3),
3. perform a projection that removes duplicate columns (A.atr2, B.atr2 → atr2, ...).

A

atr1	atr2	atr3
1	x	aa
2	y	bb
3	x	bb
4	y	bb

B

atr2	atr3	atr4	atr5
x	aa	123	11
y	bb	234	22

DATA OPERATIONS

NATURAL JOIN

A x ***B***

1

atr1	A.atr2	A.atr3	B.atr2	B.atr3	atr4	atr5
1	x	aa	x	aa	123	11
2	y	bb	x	aa	123	11
3	x	bb	x	aa	123	11
4	y	bb	x	aa	123	11
1	x	aa	y	bb	234	22
2	y	bb	y	bb	234	22
3	x	bb	y	bb	234	22
4	y	bb	y	bb	234	22

2

atr1	A.atr2	A.atr3	B.atr2	B.atr3	atr4	atr5
1	x	aa	x	aa	123	11
2	y	bb	y	bb	234	22
4	y	bb	y	bb	234	22

A►◄ B

3

atr1	atr2	atr3	atr4	atr5
1	x	aa	123	11
2	y	bb	234	22
4	y	bb	234	22

DATA OPERATIONS

EQUIJOIN

A special case of a theta join in which condition θ joins tuples having the same value for selected attributes

θ : A.atr2 = B.atr3

! Cartesian Product

A

atr1	atr2
1	x
2	y
3	x

B

atr3	atr4
x	AA
y	BB
v	CC

A►◄ ***A.atr2 = B.atr3*** **B**

atr1	atr2	atr3	atr4
1	x	x	AA
2	y	y	BB
3	x	x	AA

DATA OPERATIONS

Inner Join (INNER JOIN)

An inner join is a special case of a theta join in which condition θ joins tuples for selected attributes using a specified comparison operator (=, <>, <, >, <=, >=)

θ : A.atr2 > B.atr3

A

atr1	atr2
1	x
2	y
3	x

B

atr3	atr4
x	AA
y	BB
v	CC

A►◄ ***A.atr2 > B.atr3*** **B**

atr1	atr2	ar3	atr4
1	x	v	CC
2	y	x	AA
2	y	v	CC
3	x	v	CC

DATA OPERATIONS

Left Outer Join (LEFT OUTER JOIN)

The result of a left outer join
A LEFT OUTER JOIN B ON (join_attribute)
contains all tuples of relation A, including attribute values from relation B for tuples from B matchin

A

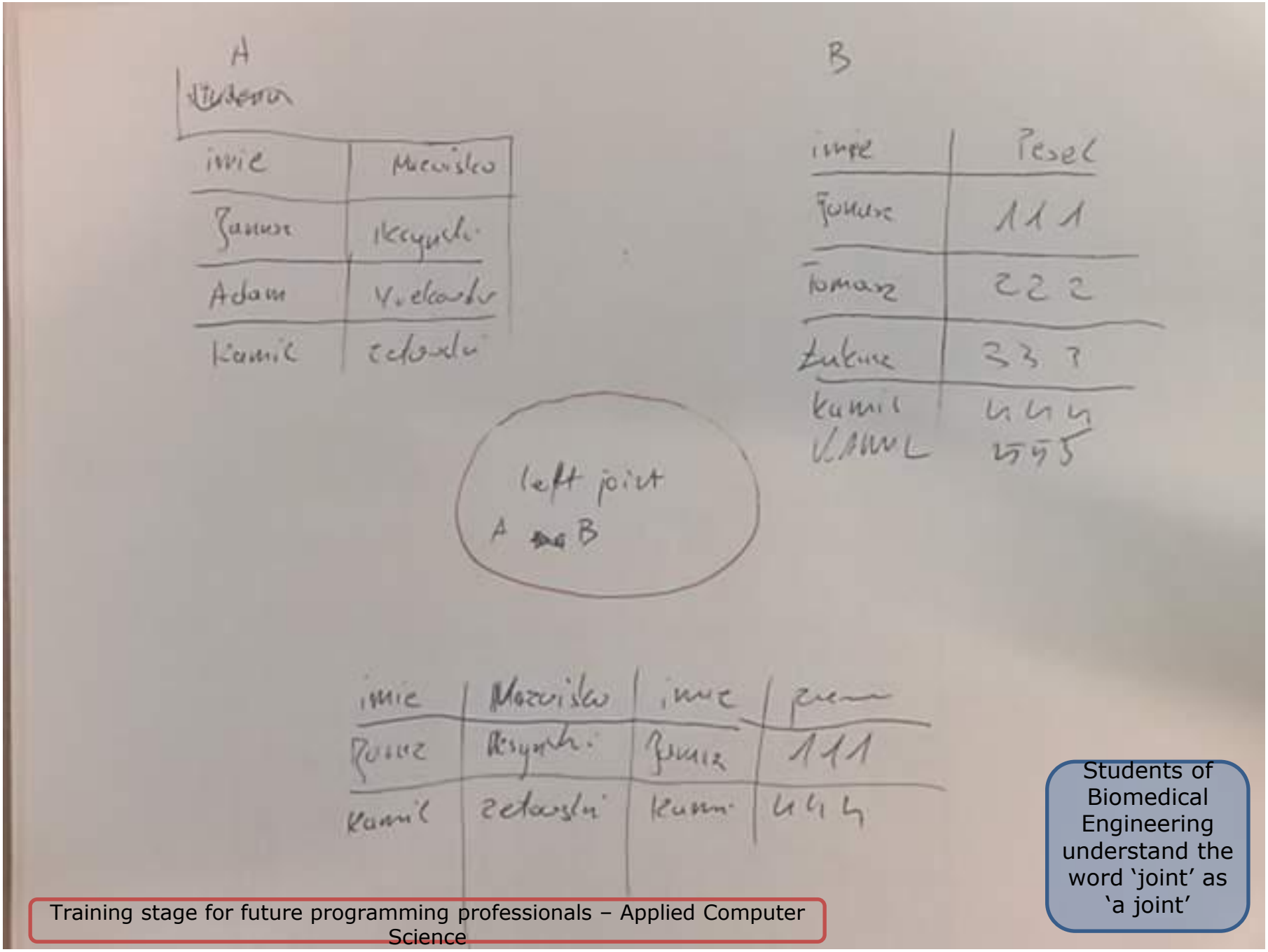
atr1	atr2
1	x
2	x
3	z

B

atr3	atr4
x	AA
y	BB
v	CC

A *LEFT OUTER JOIN* B *ON* (*A.atr2 = B.atr3*)

atr1	atr2	atr3	atr4
1	x	x	AA
2	x	x	AA
3	z		



Training stage for future programming professionals – Applied Computer Science

Students of Biomedical Engineering understand the word ‘joint’ as ‘a joint’

DATA OPERATIONS

Right Outer Join (RIGHT OUTER JOIN)

The result of a right outer join
A RIGHT OUTER JOIN B ON (join_attribute)
contains all tuples of relation B, including attribute values from relation A for tuples from A matching

A

atr1	atr2
1	x
2	x
3	z

B

atr3	atr4
x	AA
y	BB
u	CC

A *RIGHT OUTER JOIN* B *ON* (*A.atr2 = B.atr3*)

atr1	atr2	atr3	atr4
1	x	x	AA
2	x	x	AA
		y	BB
		u	CC

DATA OPERATIONS

Full Outer Join (FULL OUTER JOIN)

The result of a full outer join
A FULL OUTER JOIN B ON (join_attribute)
contains all tuples of relations A and B, including matching attribute values and null values where n

A

atr1	atr2
1	x
2	x
3	z

B

atr3	atr4
x	AA
y	BB
u	CC

A *FULL OUTER JOIN* B *ON* (*A.atr2 = B.atr3*)

atr1	atr2	atr3	atr4
1	x	x	AA
2	x	x	AA
		y	BB
		u	CC
3	z		

DATA OPERATIONS

Self-Join (self-join)

A self-join is an operation that joins data from the same table.

Nr	Imię	Nazwisko	Nr szefa
1	Jerzy	Nowak	
2	Anna	Wójcik	1
3	Aneta	Mazur	2



Nr	Imię	Nazwisko	Sz.Imię	Sz.Nazw
1	Jerzy	Nowak		
2	Anna	Wójcik	Jerzy	Nowak
3	Aneta	Mazur	Anna	Wójcik

Semi-Join

A semi-join is a join of two tables in which the result contains data from only one table. It is used when decomposing nested queries or restricting solutions.

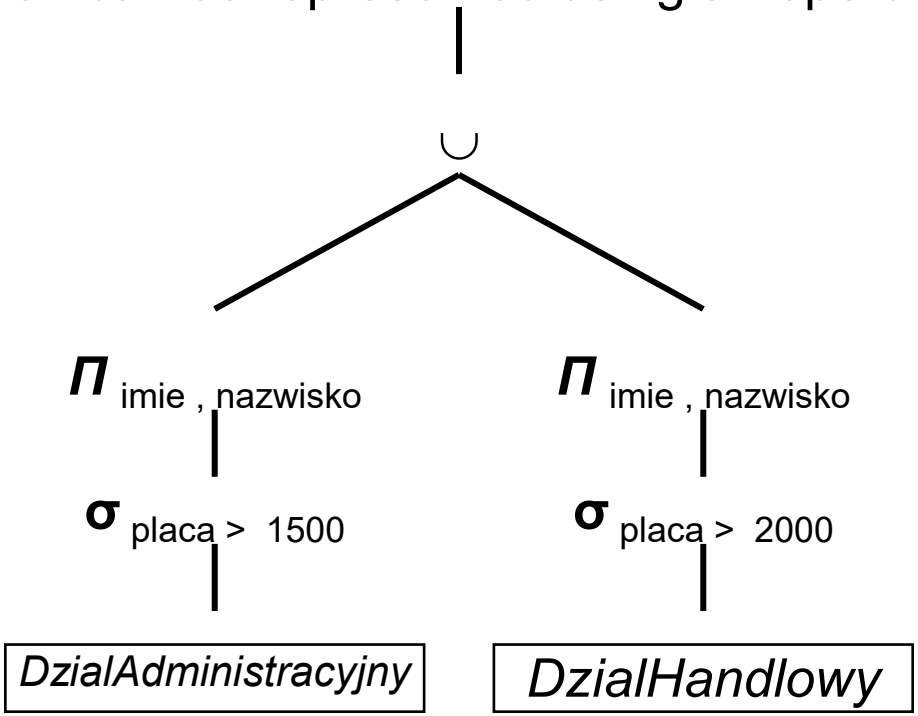
DATA OPERATIONS

COMPOSITIONS

Relational algebra allows operations to be composed. The order of execution can be specified by the operator ().

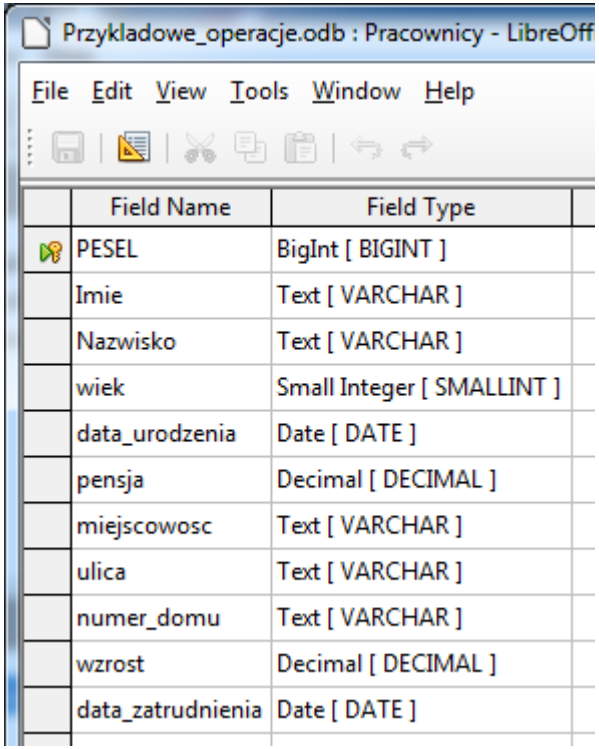
(Π IMIE, NAZWISKO (σ PLACA > 1500 (DzialAdministracyjny))) ∪ (Π IMIE, NAZWISKO (σ PLACA > 2000 (DzialHandlowy)))

The above composition can be represented using an operation tree:

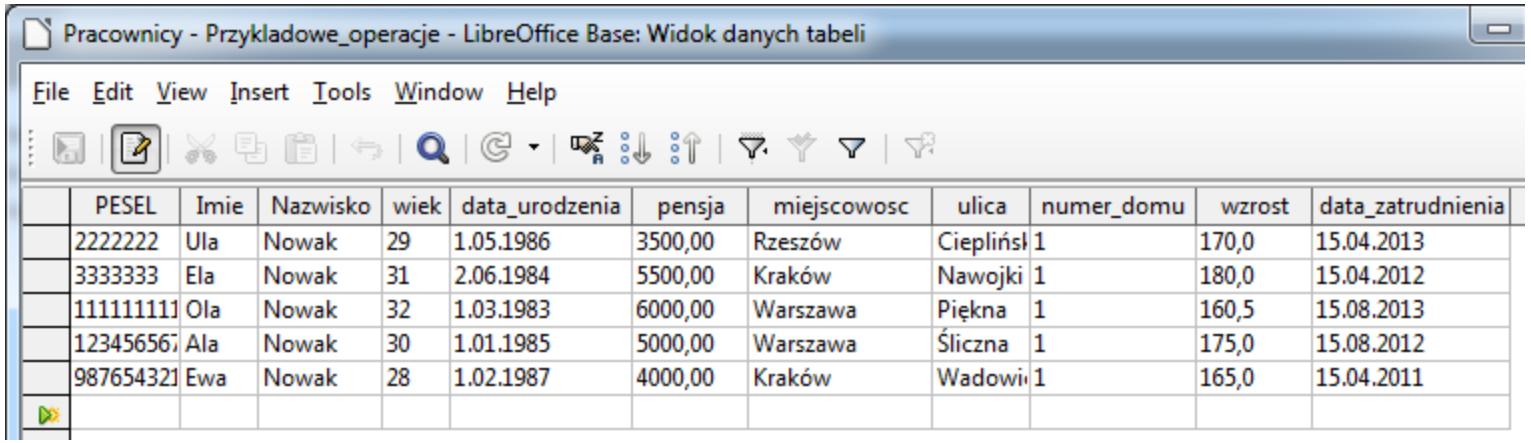


Example Queries in LibreOffice

Example table with data

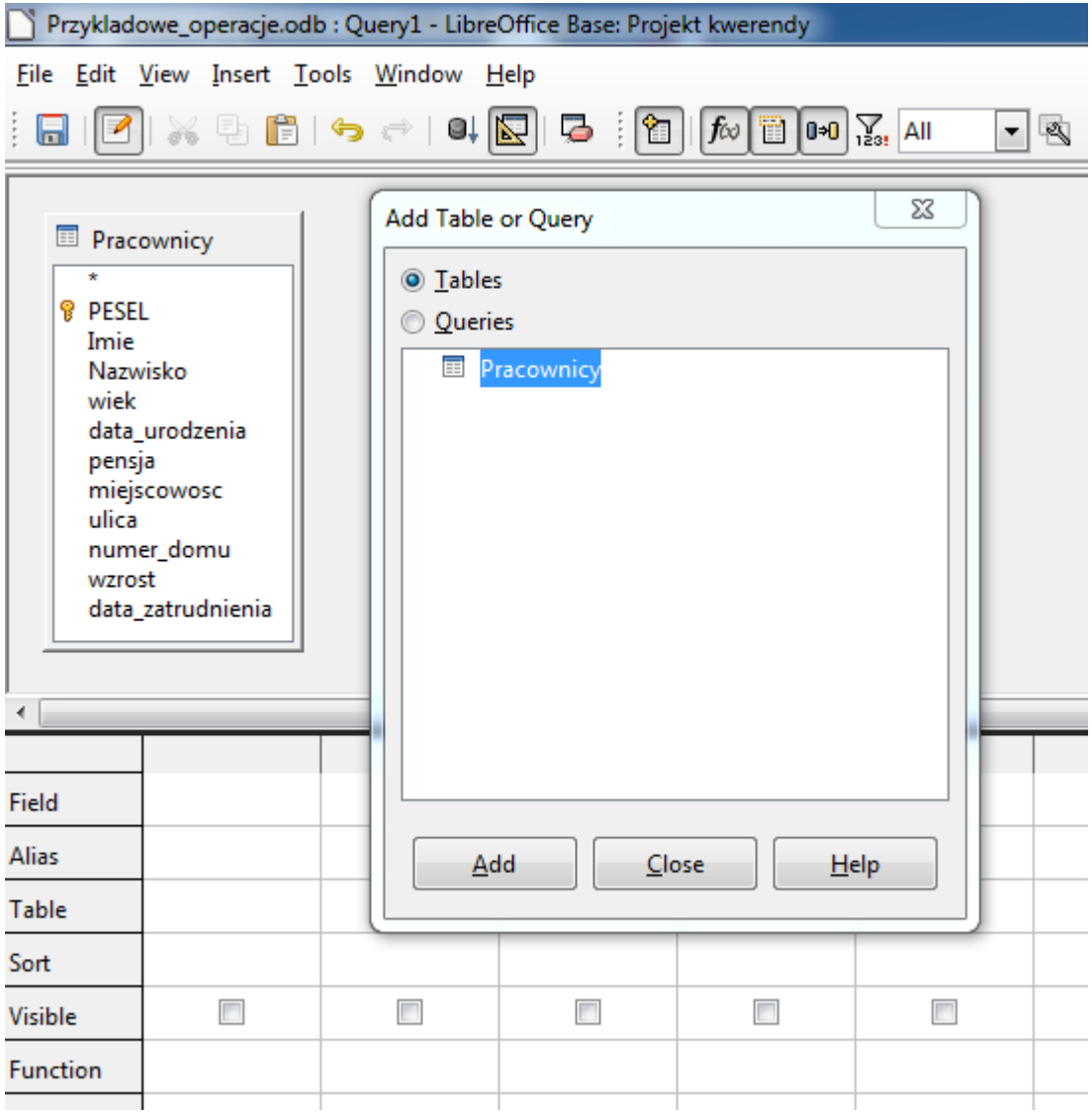


Field Name	Field Type
PESEL	BigInt [BIGINT]
Imie	Text [VARCHAR]
Nazwisko	Text [VARCHAR]
wiek	Small Integer [SMALLINT]
data_urodzenia	Date [DATE]
pensja	Decimal [DECIMAL]
miejscowosc	Text [VARCHAR]
ulica	Text [VARCHAR]
numer_domu	Text [VARCHAR]
wzrost	Decimal [DECIMAL]
data_zatrudnienia	Date [DATE]



PESEL	Imie	Nazwisko	wiek	data_urodzenia	pensja	miejscowosc	ulica	numer_domu	wzrost	data_zatrudnienia
2222222	Ula	Nowak	29	1.05.1986	3500,00	Rzeszów	Ciepliński	1	170,0	15.04.2013
3333333	Ela	Nowak	31	2.06.1984	5500,00	Kraków	Nawojki	1	180,0	15.04.2012
111111111	Ola	Nowak	32	1.03.1983	6000,00	Warszawa	Piękna	1	160,5	15.08.2013
12345656	Ala	Nowak	30	1.01.1985	5000,00	Warszawa	Śliczna	1	175,0	15.08.2012
987654321	Ewa	Nowak	28	1.02.1987	4000,00	Kraków	Wadowi	1	165,0	15.04.2011

Example Queries in LibreOffice

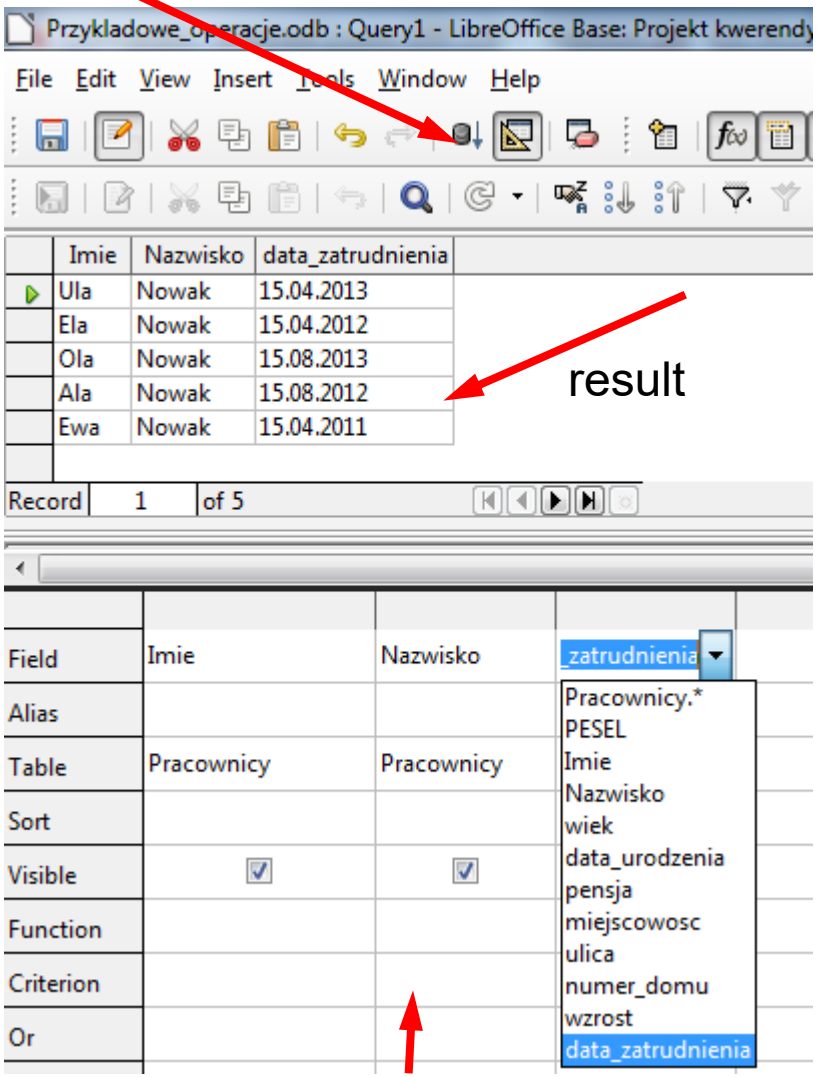


Selecting tables participating in the query – initially, one table

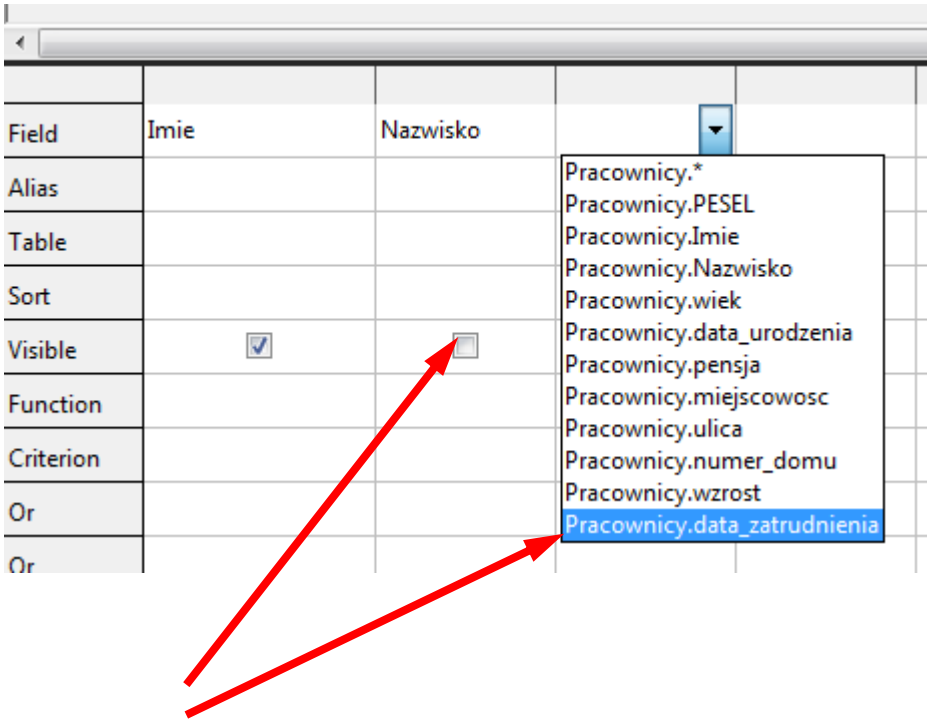
Example Queries in LibreOffice

run query

PROJECTION



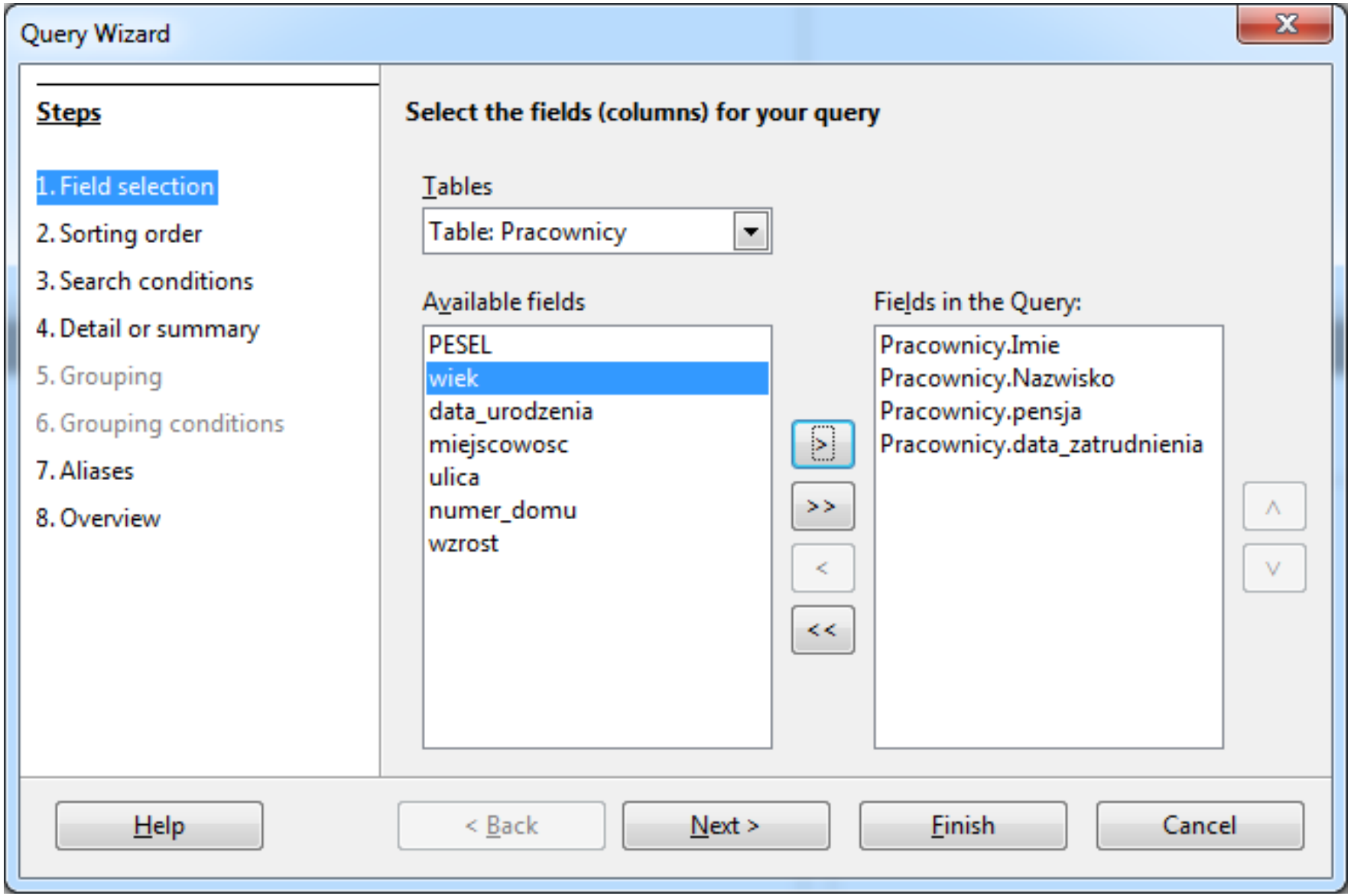
result



field selection

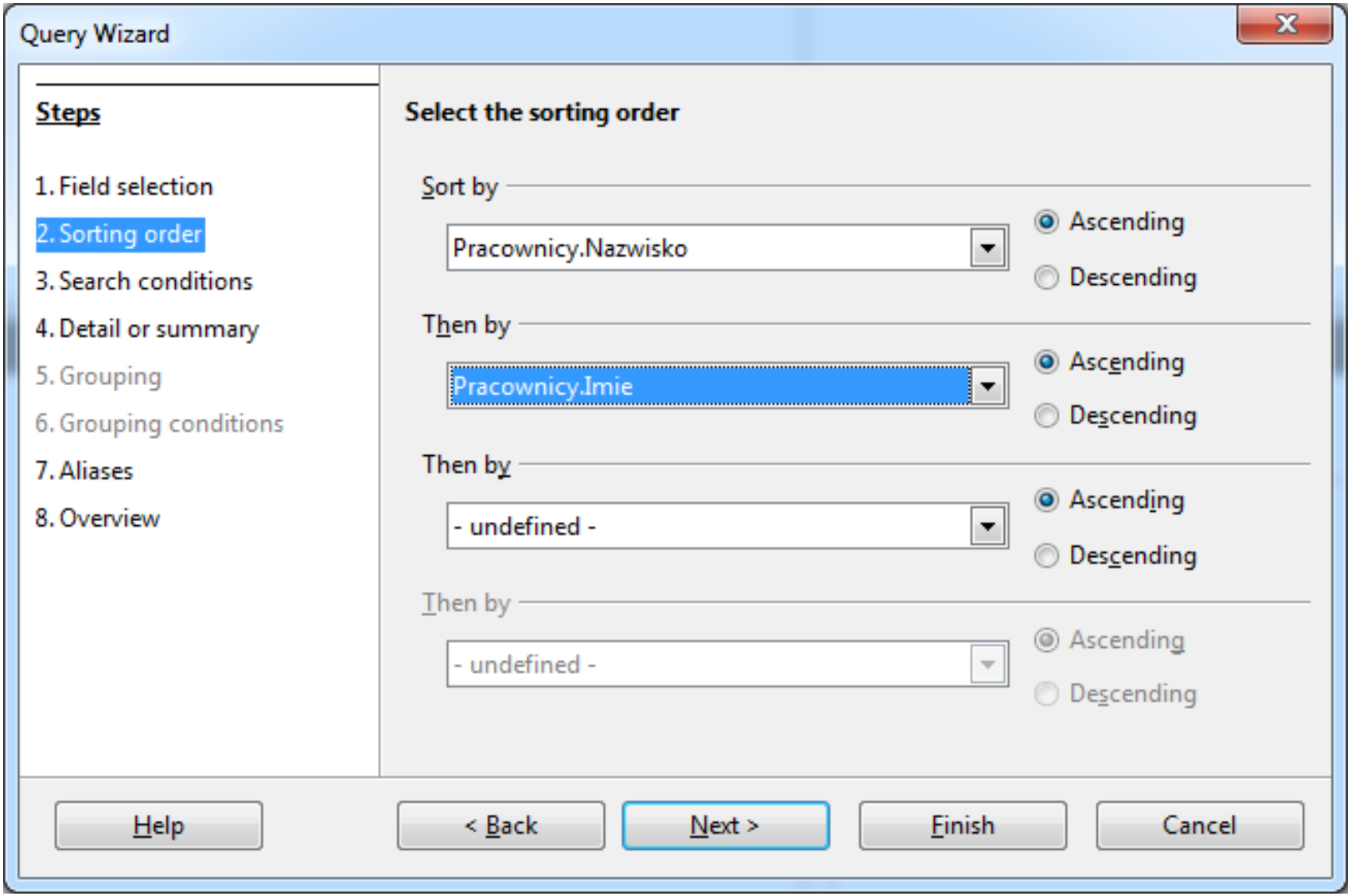
Query Design

Example Queries in LibreOffice



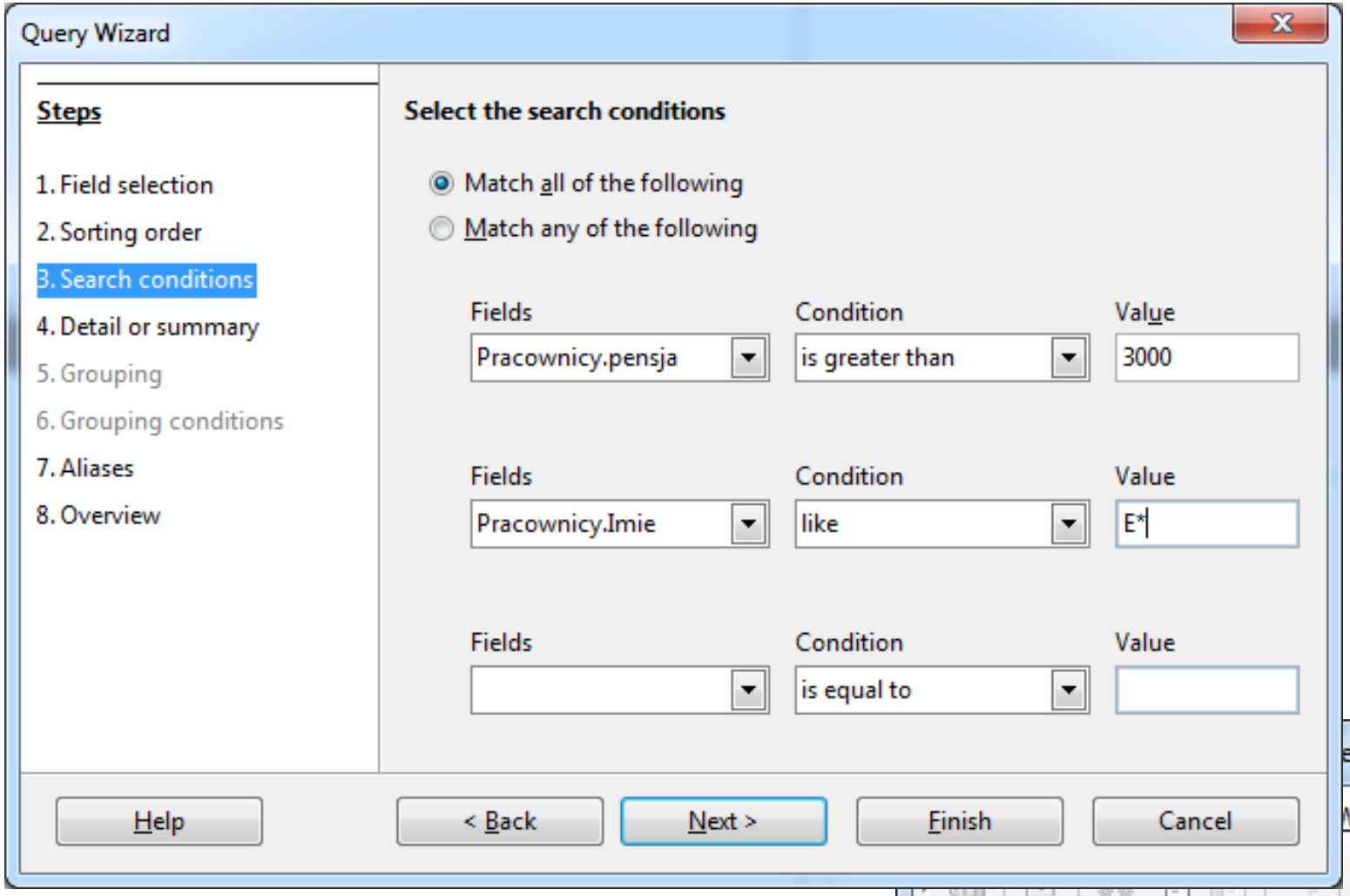
Another method: using the so-called wizard

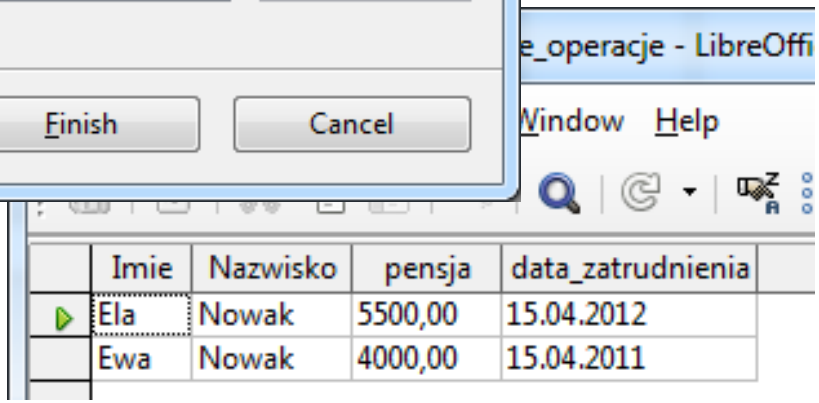
Example Queries in LibreOffice



data sorting

Example Queries in LibreOffice

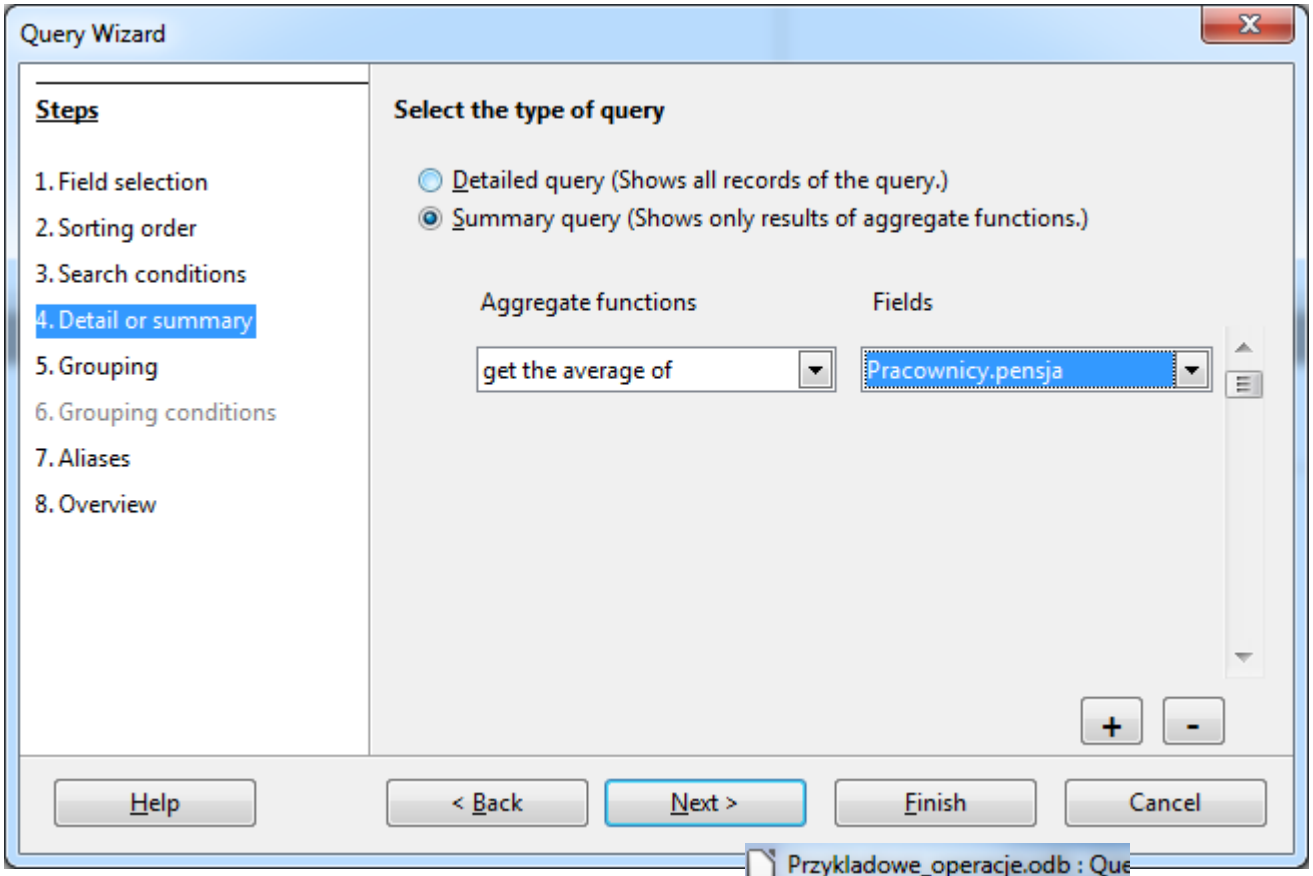




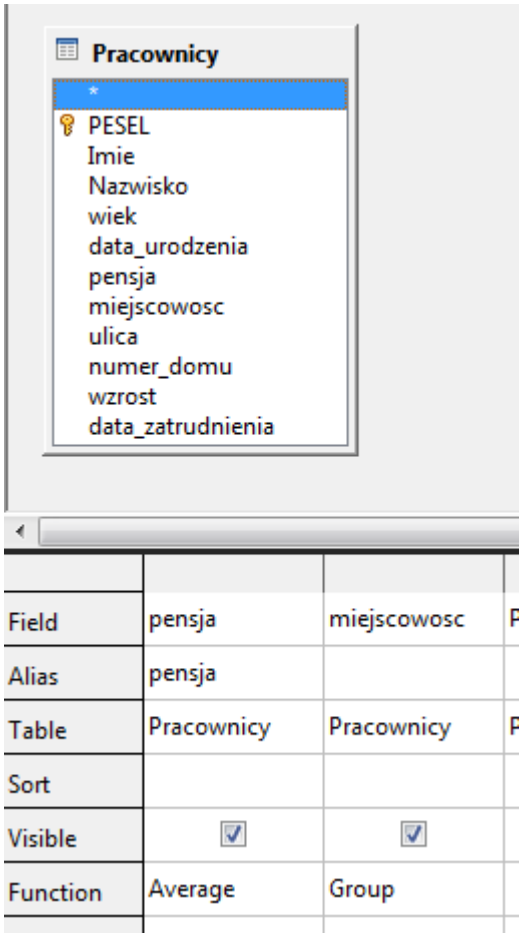
Entering example criteria
for numbers (roomry) and text (first name)

Example Queries in LibreOffice

GROUPING

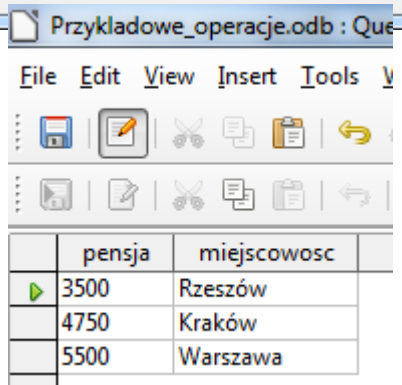


wizard



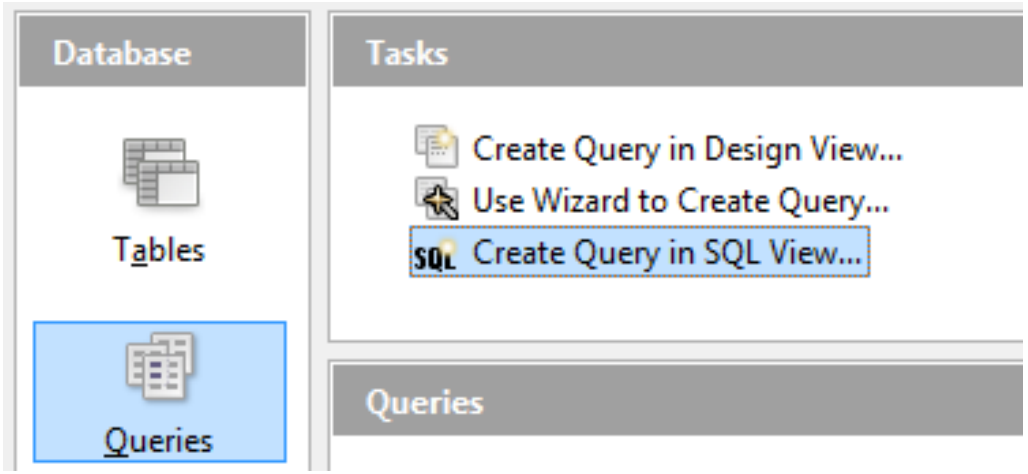
design

result



Example Queries in LibreOffice

GROUPING



Third method of creating queries

