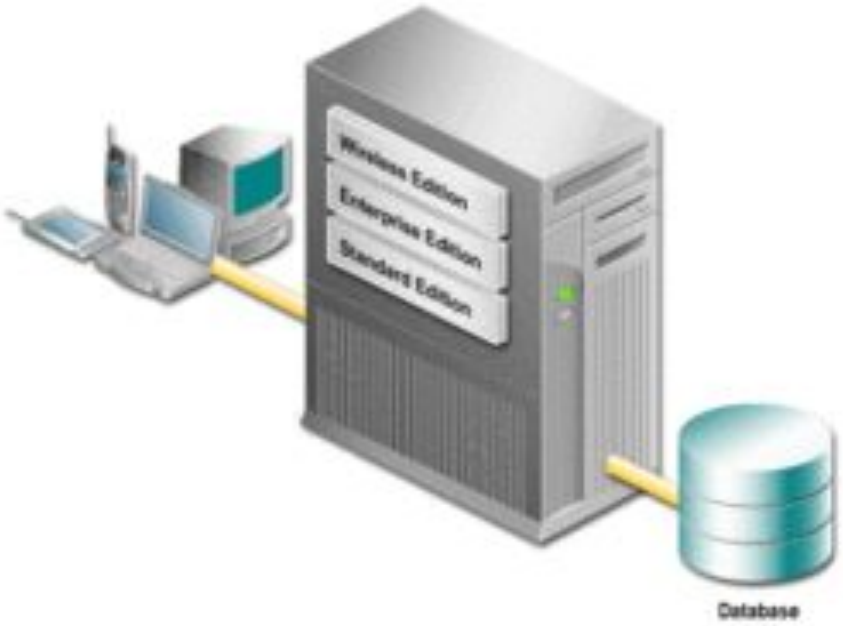
Bazy danych
Administracja



Opracował:
Artem Nowicki

Przypomnienie pojęć

DBMS - systemem zarządzania bazą danych. W bazie danych dane są organizowane przez ten właśnie system.

Po polsku ten system nazywany jest **System zarządzania bazą danych,** czyli **SZBD**

Tabela PRACOWNIK

Nazwisko
Imię
PESEL
NIP

Tabela WYNAGRODZENIE

Data
Kwota
Tytuł
NIP

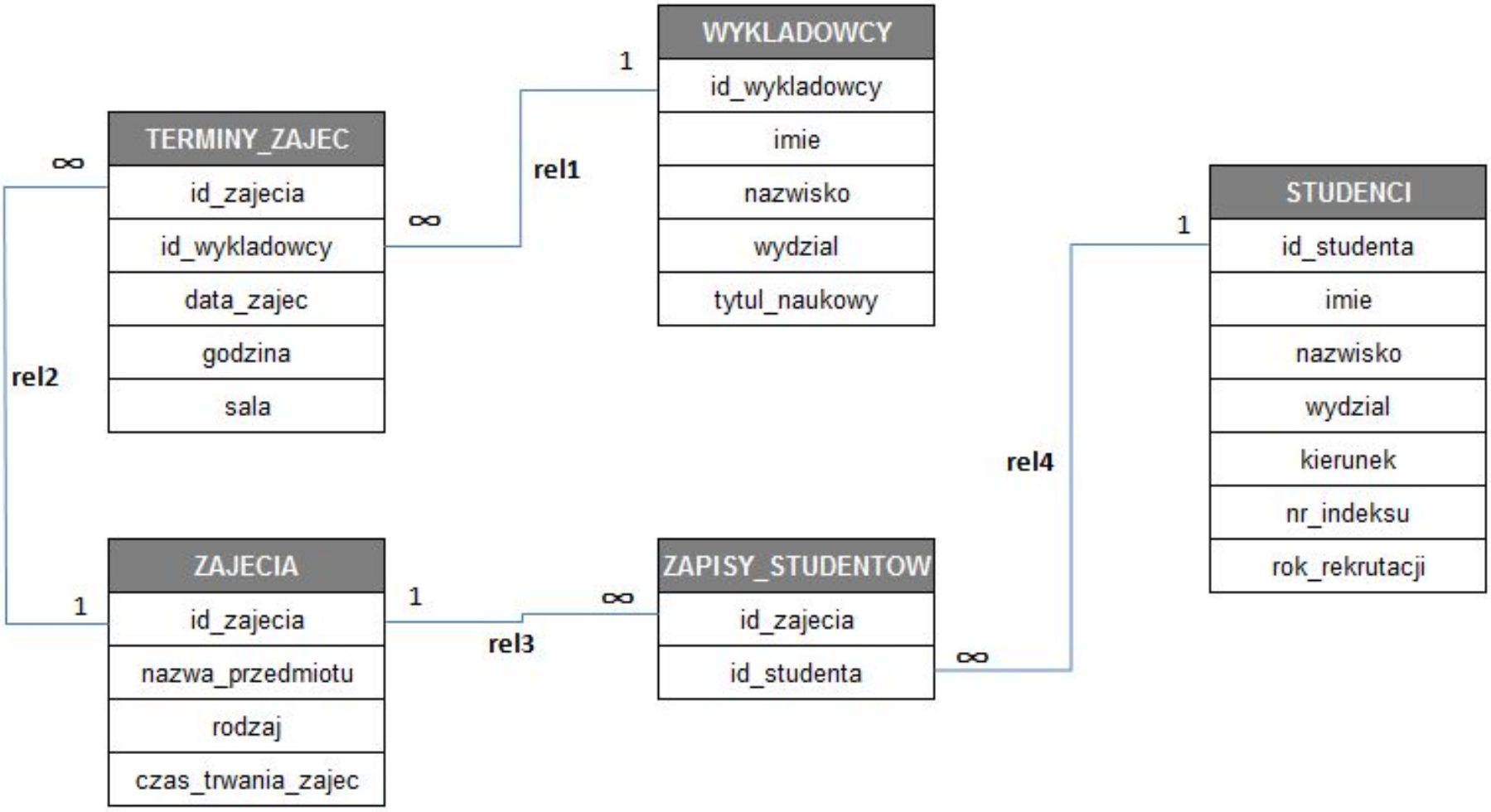
24-06-2005	1850,00	wyangrodzenie	123-12-12-123

Kowalski	Jan	65123112345	123-12-12-123

W 1970 E. F. Codd zaproponował relacyjny model danych.

Relacje - reprezentowane są przez tablice.

Relacje są pewnym zbiorem rekordów o identycznej strukturze wewnętrznie powiązanych za pomocą związków zachodzących pomiędzy danymi.

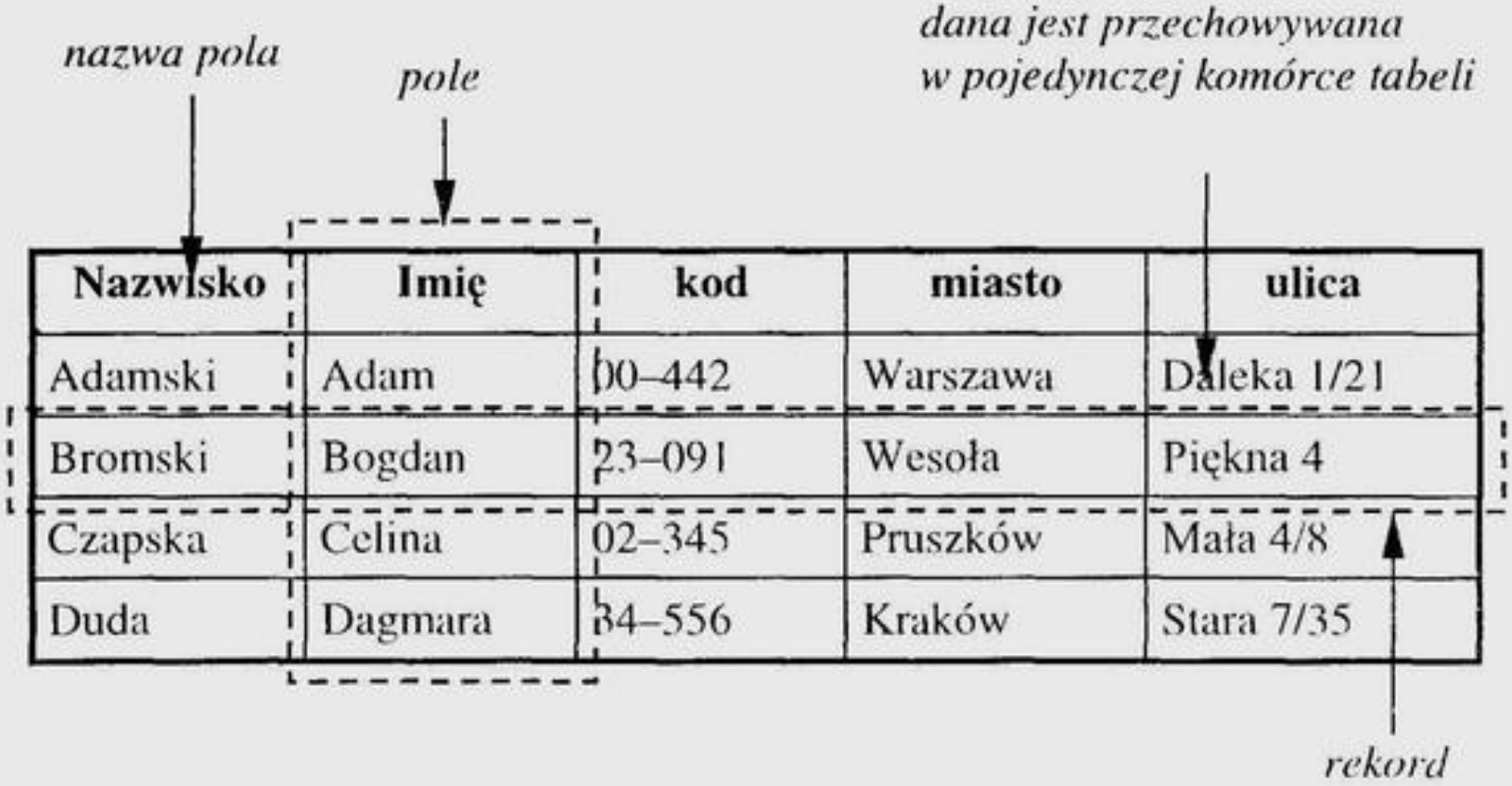

WYKLADOWCY
id_wykladowcy
imie
nazwisko
wydzial
tytul_naukowy
TERMINY_ZAJEC
id_zajecia
id_wykladowcy
data_zajec
godzina
sala
STUDENCI
id_studenta
imie
nazwisko
wydzial
kierunek
nr_indeksu
rok_rekrutacji
ZAJECIA
id_zajecia
nazwa_przedmiotu
rodzaj
czas_trwania_zajec
ZAPISY_STUDENTOW
id_zajecia
id_studenta
rel1
rel2
rel3
rel4
1
∞

tabela

Pojedyncza **tabela** może być reprezentacją pewnej encji (np. książek, mieszkań, ludzi), relacji między nimi, albo może stanowić zawartość całej **bazy danych**.

Pojedynczy wiersz **tabeli** nazywany jest rekordem i stanowi najczęściej zbiór **danych** o pojedynczym obiekcie

ATRYBUT w praktyce, to nic innego jako KOLUMNA



Nazwisko	Imię	kod	miasto	ulica
Adamski	Adam	00–442	Warszawa	Daleka 1/21
Bromski	Bogdan	23–091	Wesoła	Piękna 4
Czapska	Celina	02–345	Pruszków	Mała 4/8
Duda	Dagmara	34–556	Kraków	Stara 7/35

Rysunek 1. Organizacja tabeli

Kolumny

Każda **KOLUMNA** jest ściśle określona TYPEM DANYCH, czyli przechowuje wartości jednorodne, z określonej DZIEDZINY (tego samego typu np. liczby, znaki, daty etc).

Nazwa kolumny w ramach tabeli musi być unikalna, bo silnik musi jednoznacznie wiedzieć do którego atrybutu będziemy się odnosić.

Encja

Encja (ang. entity) – reprezentacja wyobrażonego lub rzeczywistego obiektu (grupy obiektów) stosowana przy modelowaniu danych podczas analizy informatycznej.

Pojęcie to posiada różniące się definicje, zmieniające się z czasem.

Encja wg. R. Barkera

W tym podejściu modelu encji w logiczny schemat bazy danych **encja to tablica** (tabela), **atrybuty encji to kolumny tabeli**, a zatem wystąpienie encji to rekord (wiersz) tabeli.

Barker zaleca, aby dla nazw encji używać liczby mnogiej (np. "Samoloty"), a dla jej wystąpień - liczby pojedynczej ("Samolot).

Krotka

To pojedynczy egzemplarz, czyli obiekt opisany wszystkimi ATRYBUTAMI danej RELACJI.
KROTKA to nic innego jak WIERSZ czy REKORD.

Encja i wystąpienie encji

Obiekty świata rzeczywistego

Firma zatrudnia pracowników. Chcemy przechowywać informacje nt. danych personalnych pracowników (imię, nazwisko, adres i numer telefonu).

John Smith
Long Street 23
phone 253-485

Andy Green
White Street 17
phone 333-951

Eva Brown
Black Street 5/2
phone 753-624

Identyfikacja wspólnych cech obiektów

Model

Encja

Employee
first_name
last_name
address
telephone

Wystąpienia encji

Employee
first name = John
last name = Smith
address = Long Street 23
telephone = 256-485

Reguły dotyczące encji:

- Encja jest zbiorem obiektów nazywanych instancjami encji
- Nazwa encji powinna być rzeczownikiem w l. pojedynczej
- Dowolna rzecz lub obiekt może być reprezentowana tylko przez jedną encję; każda encja musi być jednoznacznie identyfikowalna

KLUCZE

Klucze to zbiory atrybutów mających określoną właściwość. Dzięki nim, możemy jednoznacznie identyfikować każdy pojedynczy wiersz.

Znajomość pojęć kluczy podstawowych i obcych jest niezbędna do tworzenia zapytań, odwołujących się do wielu tabel.

SUPERKLUCZ (NADKLUCZ)

Superkluczem nazywamy dowolny podzbiór atrybutów, identyfikujący jednoznacznie każdy wiersz. Każda RELACJA (tabela) może zawierać wiele takich kluczy.

Szczególnym przypadkiem jest superklucz składający się ze wszystkich atrybutów (kolumn) danej tabeli.

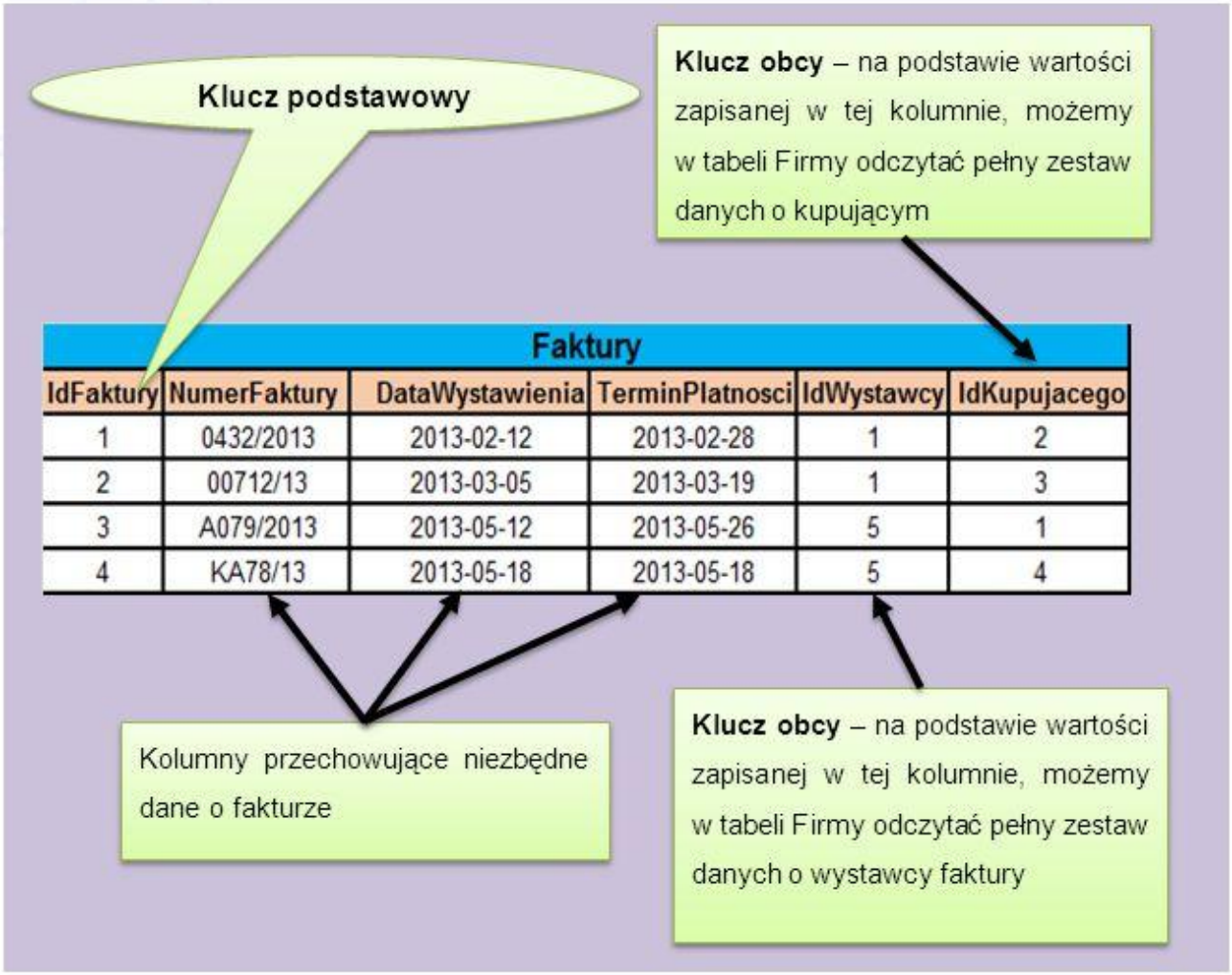
KLUCZ PODSTAWOWY (PRIMARY KEY)

To wybrany (zazwyczaj najkrótszy), jednoznacznie identyfikujący każdy, pojedynczy wiersz, zbiór atrybutów (kolumn) danej relacji (tabeli).

W praktyce, będzie to najczęściej jedna lub dwie kolumny w tabeli, jednoznacznie (UNIKALNIE) identyfikujący każdy wiersz. Nie można stworzyć klucza podstawowego, na zbiorze atrybutów nieunikalnych.

Dwa wiersze nie mogą mieć takiej samej wartości klucza podstawowego.

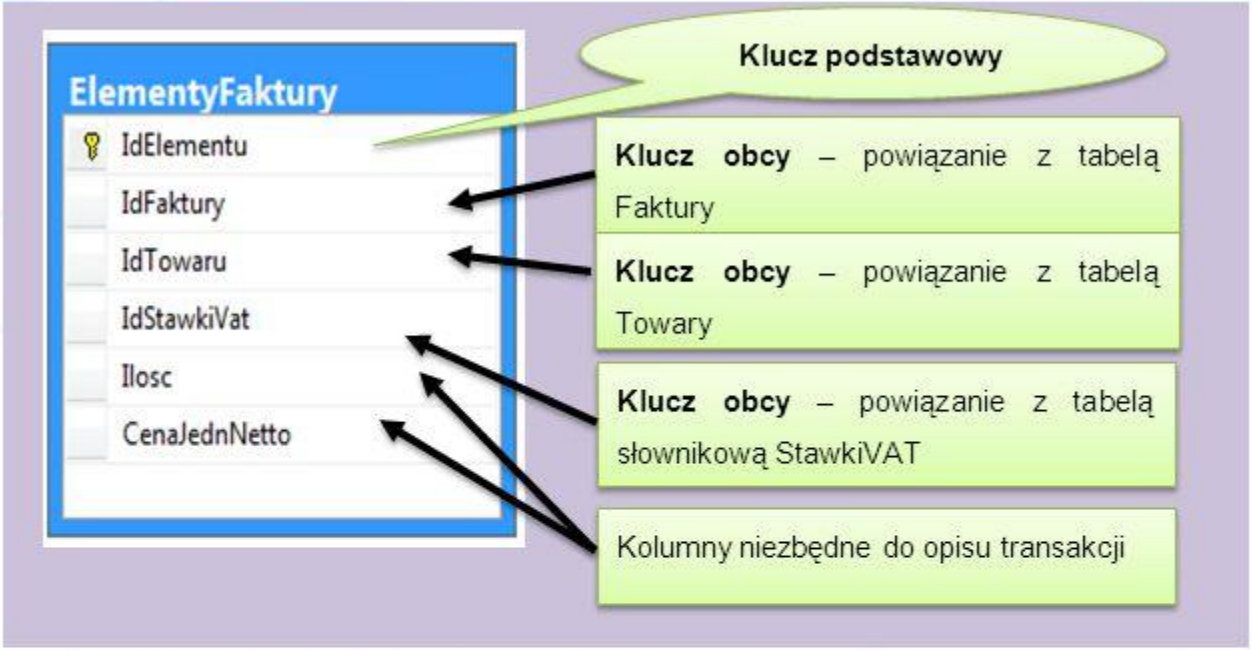
Projekt bazy danych – normalizacja

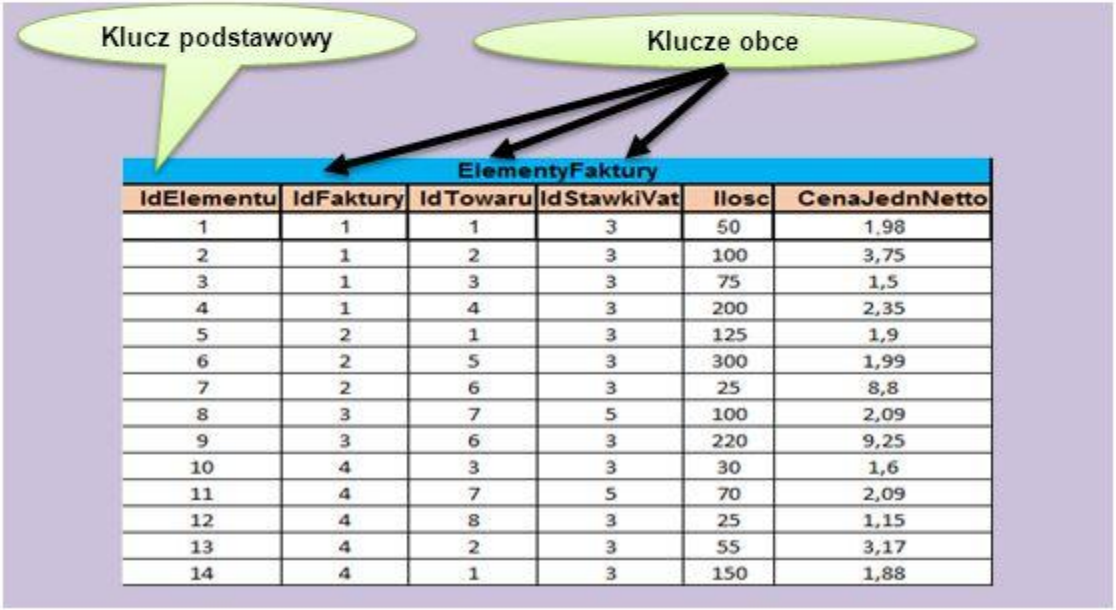


Faktury

IdFaktury	NumerFaktury	DataWystawienia	TerminPlatnosci	IdWystawcy	IdKupujacego
1	0432/2013	2013-02-12	2013-02-28	1	2
2	00712/13	2013-03-05	2013-03-19	1	3
3	A079/2013	2013-05-12	2013-05-26	5	1
4	KA78/13	2013-05-18	2013-05-18	5	4

Projekt bazy danych – normalizacja





ElementyFaktury

IdElementu	IdFaktury	IdTowaru	IdStawkiVat	Ilosc	CenaJednNetto
1	1	1	3	50	1,98
2	1	2	3	100	3,75
3	1	3	3	75	1,5
4	1	4	3	200	2,35
5	2	1	3	125	1,9
6	2	5	3	300	1,99
7	2	6	3	25	8,8
8	3	7	5	100	2,09
9	3	6	3	220	9,25
10	4	3	3	30	1,6
11	4	7	5	70	2,09
12	4	8	3	25	1,15
13	4	2	3	55	3,17
14	4	1	3	150	1,88

KLUCZ NATURALNY i SZTUCZNY

- **Kluczem naturalnym**, będzie kolumna (lub zbiór kolumn) opisująca daną klasę obiektów – np. NIP. Jest to atrybut, który z punktu widzenia systemu postrzegany jest tak samo naturalnie jak Nazwa firmy czy jej REGON.

- Może być adres email użytkownika systemu. Przeważnie zakładamy, że dwóch użytkowników nie może mieć takiego samego adresu.

Klucz sztuczny

- kolumna stworzona przez projektanta bazy danych w celu identyfikacji rekordów, możliwie krótkim kluczem.

- Zazwyczaj będzie to wartość liczbowa typu całkowitego (INT, SMALLINT, BIGINT).

KLUCZ OBCY

- To atrybut lub zbiór atrybutów, wskazujący na KLUCZ GŁÓWNY w innej RELACJI (tabeli). Klucz obcy to nic innego jak związek, relacja między dwoma tabelami.
- Cecha dobrego klucza głównego (możliwie krótki) tutaj staje się klarowna. W tabeli powiązanej kluczem obcym, trzeba powielić tą strukturę (zbiór atrybutów) aby móc jednoznacznie wiązać rekordy z dwóch tabel.
- Definicja klucza obcego - pilnuje aby w tabeli powiązanej, w określonych atrybutach, znaleźć się mogły tylko takie wartości które istnieją w tabeli docelowej jako klucz główny. Klucz obcy może dotyczyć również tej samej tabeli.

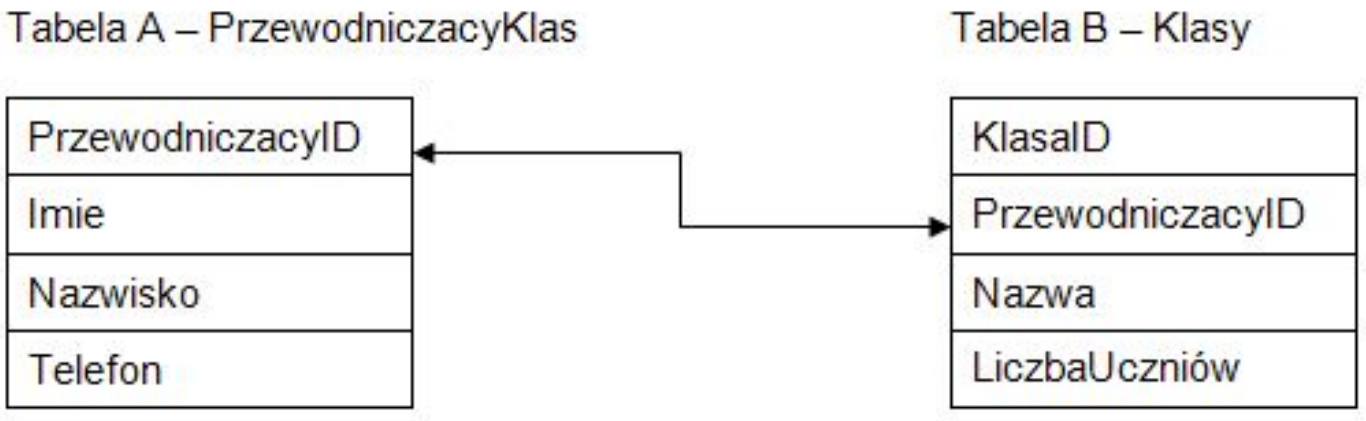
Powiązania pomiędzy tabelami

W praktyce spotkać możemy trzy fundamentralne związki między tabelami.

Dzięki nim, możemy zapewnić integralność referencyjną danych i zamodelować odpowiednią logikę naszej struktury.

ZWIĄZEK 1:1 (jeden do jeden)

Każdy wiersz z tabeli A może mieć tylko jednego odpowiednika w tabeli B (i na odwrót)
Ten rodzaj relacji może być postrzegany jako podzielenie tabeli na dwie (bo relacja jest jeden do jeden). Stosowany np. wtedy, gdy zbiór dodatkowych atrybutów jest określony tylko dla wąskiego podzbioru wierszy w tabeli podstawowej.



1:1

Innym zastosowaniem związku 1:1, jest wydzielenie pewnej grupy atrybutów które są rzadko odpytywane. Mogą być, więc umiejscowione w tabeli przechowywanej na osobnym wolniejszym, nośniku danych.

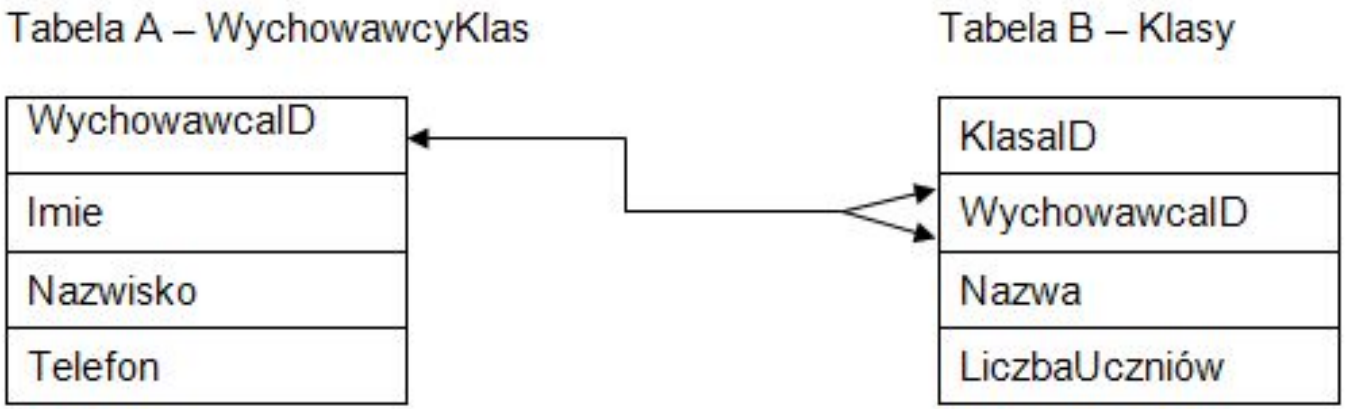
Kolejny scenariusz to dodatkowa ochrona części atrybutów określonego typu (np. informacji wrażliwych takich jak wynagrodzenie, preferencje etc.).

Wydzielając je do osobnej tabeli, możemy zapewnić dodatkowy poziom zabezpieczeń (dostęp, szyfrowanie), inną politykę backupową etc..

ZWIĄZEK 1:N (jeden do wiele)

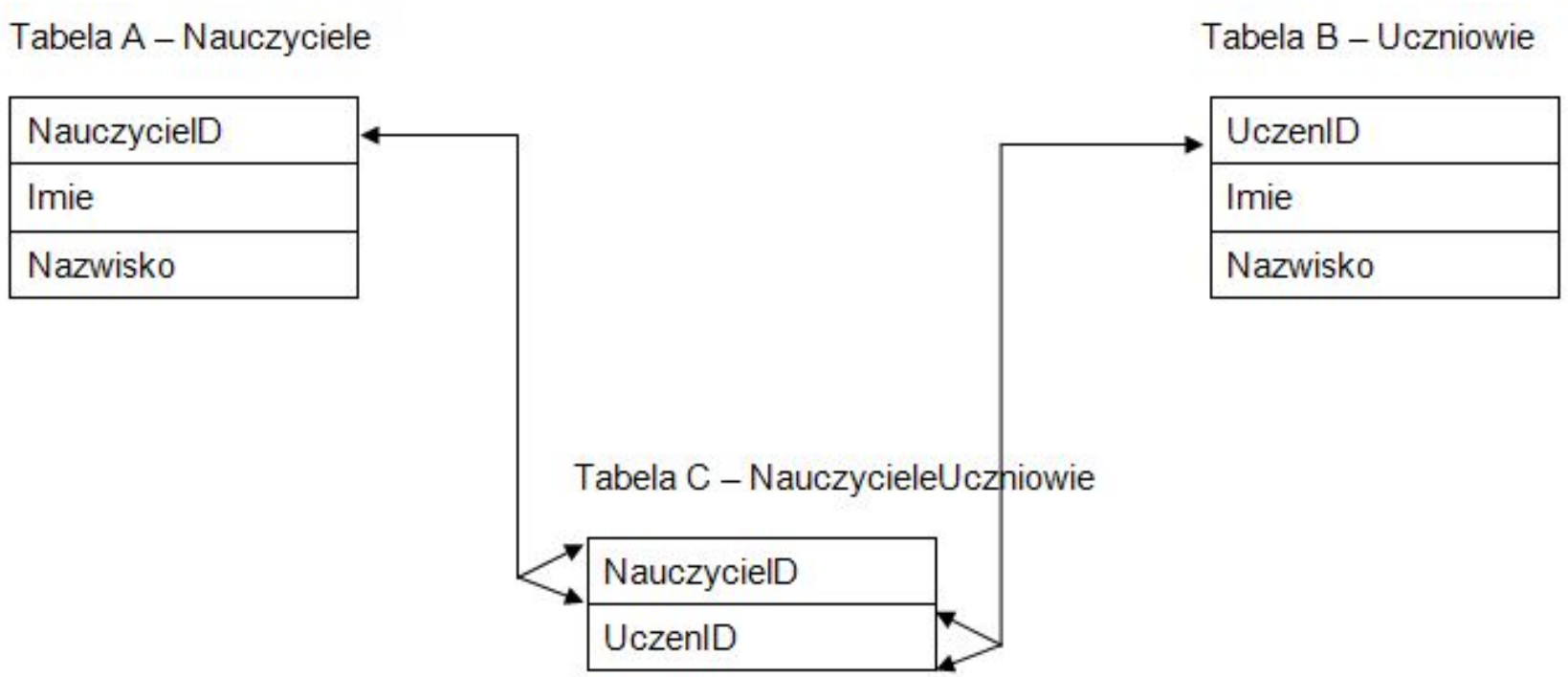
Jest to najczęściej spotykana relacja. Określamy w niej że każdy element ze zbioru A (wiersz tabeli A), może być powiązany z wieloma elementami zbioru B.

W tym modelu wiele produktów może należeć do jednej kategorii.



ZWIĄZEK N:M (wiele do wiele)

Realizowana jest zawsze jako dwie relacje 1:N. Zatem jeśli chcemy między dwoma tabelami zamodelować związek N:M **potrzebujemy trzecią tabelę – łącznikową**.



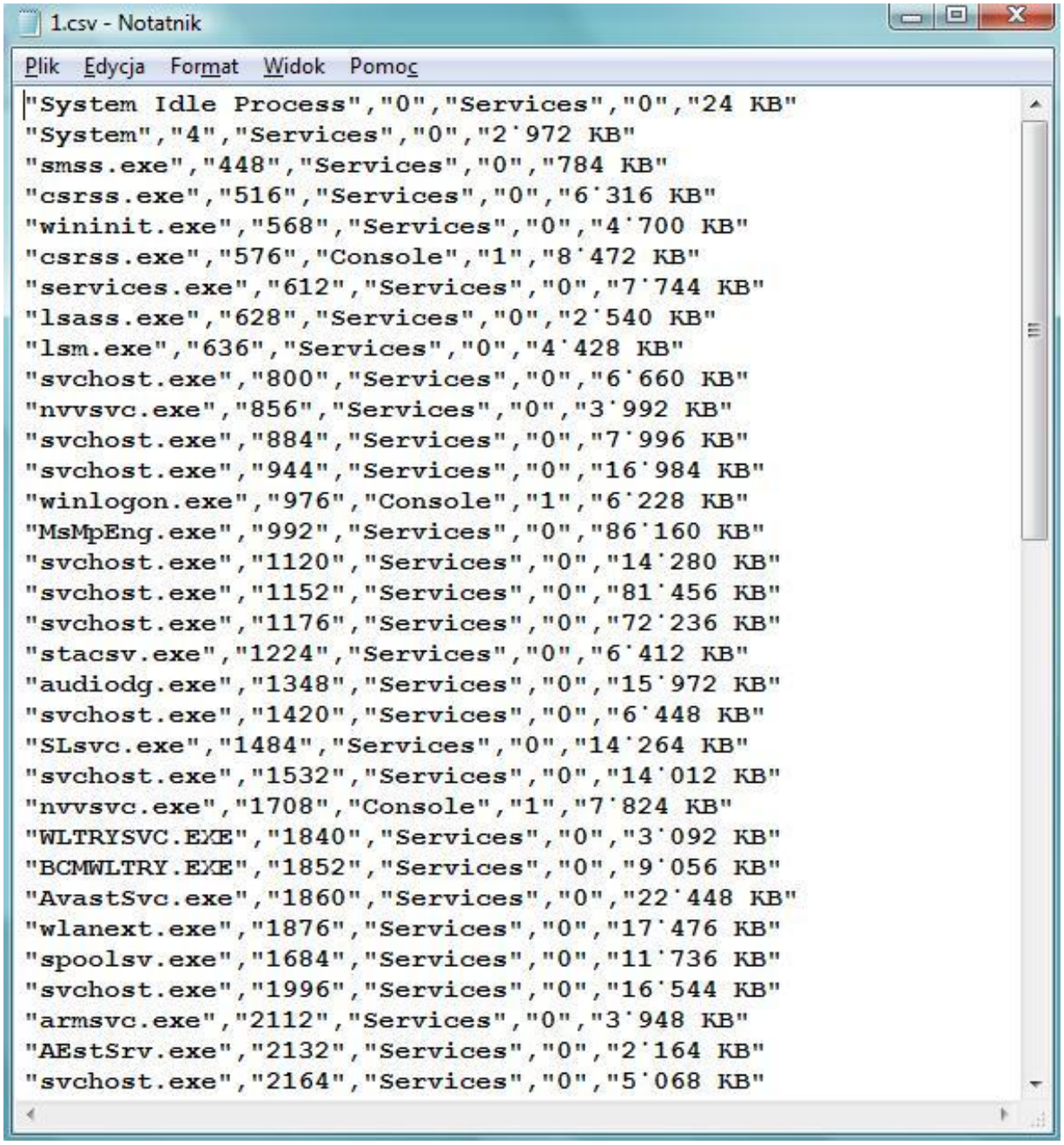
Podstawowe bazy danych

Bazę danych można utworzyć na wiele sposobów. Najprostsze rozwiązania to zwykły plik tekstowy z utworzonymi strukturami pól i rekordów rozdzielone średnikami lub innymi umownymi znakami.

Dostęp do takiego pliku można uzyskać np. z poziomu języka bash w Linux. Komendy: grep

W UNIX/Linux rozszerzenia plików nie mają znaczenia ale w Windows pliki takie mają atrybut ***.csv.**

Import do Excel tworzy tabelę.


1.csv - Notatnik
Plik Edycja Format Widok Pomoc
"System Idle Process","0","Services","0","24 KB"
"System","4","Services","0","2˙972 KB"
"smss.exe","448","Services","0","784 KB"
"csrss.exe","516","Services","0","6˙316 KB"
"wininit.exe","568","Services","0","4˙700 KB"
"csrss.exe","576","Console","1","8˙472 KB"
"services.exe","612","Services","0","7˙744 KB"
"lsass.exe","628","Services","0","2˙540 KB"
"lsm.exe","636","Services","0","4˙428 KB"
"svchost.exe","800","Services","0","6˙660 KB"
"nvvsvc.exe","856","Services","0","3˙992 KB"
"svchost.exe","884","Services","0","7˙996 KB"
"svchost.exe","944","Services","0","16˙984 KB"
"winlogon.exe","976","Console","1","6˙228 KB"
"MsMpEng.exe","992","Services","0","86˙160 KB"
"svchost.exe","1120","Services","0","14˙280 KB"
"svchost.exe","1152","Services","0","81˙456 KB"
"svchost.exe","1176","Services","0","72˙236 KB"
"stacsv.exe","1224","Services","0","6˙412 KB"
"audiodg.exe","1348","Services","0","15˙972 KB"
"svchost.exe","1420","Services","0","6˙448 KB"
"SLsvc.exe","1484","Services","0","14˙264 KB"
"svchost.exe","1532","Services","0","14˙012 KB"
"nvvsvc.exe","1708","Console","1","7˙824 KB"
"WLTRYSVC.EXE","1840","Services","0","3˙092 KB"
"BCMWLTRY.EXE","1852","Services","0","9˙056 KB"
"AvastSvc.exe","1860","Services","0","22˙448 KB"
"wlanext.exe","1876","Services","0","17˙476 KB"
"spoolsv.exe","1684","Services","0","11˙736 KB"
"svchost.exe","1996","Services","0","16˙544 KB"
"armsvc.exe","2112","Services","0","3˙948 KB"
"AEstSrv.exe","2132","Services","0","2˙164 KB"
"svchost.exe","2164","Services","0","5˙068 KB"

Systemy baz danych

Systemy obsługi relacyjnych baz danych wchodzą w skład praktycznie każdego pakietu biurowego:

- Ms Office
- OpenOffice.org
- LibreOffice

Język SQL

SQL (ang. *Structured Query Language)*

Język SQL służy do opisywania zbiorów danych umożliwiających uzyskiwanie odpowiedzi na pytania.

Opracowany w latach 70. w firmie IBM. Stał się standardem w komunikacji z serwerami relacyjnych baz danych.

Pierwszą firmą, która włączyła SQL do swojego produktu komercyjnego, był Oracle.

SQL cd..

- Rozpoznawany jest przez wszystkie najpopularniejsze systemy baz danych takie jak np.
- MySQL,
- PostgreSQL,
- Microsoft SQL Server,
- Oracle,
- DB2.

MySQL rozwijany jest przez firmę Oracle.
MySQL AB została kupiona 16 stycznia 2008 roku przez Sun Microsystems, a ten 27 stycznia 2010 roku przez Oracle.

W międzyczasie Monty Widenius (współtwórca MySQL) stworzył **MariaDB** (alternatywną wersję) opartego na licencji GPL.

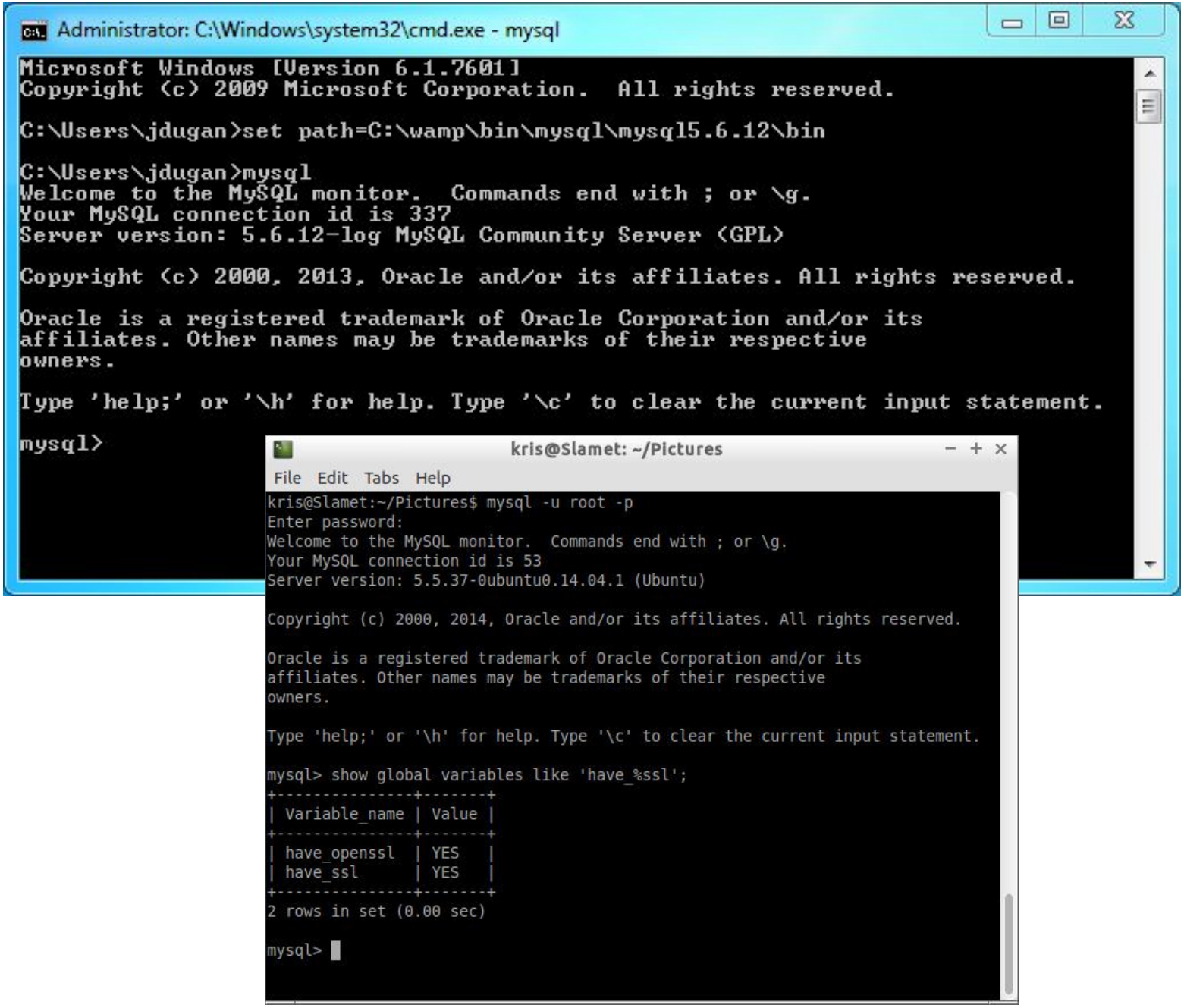
MariaDB jest oparta na tym samym kodzie bazowym co MySQL i dąży do utrzymania kompatybilności z jej poprzednimi wersjami

MariaDB

- Kompatybilność z MySql (różnie)
- Szybkość działania / wydajność
- Społeczność
- Wsparcie - CentOS 7 (już wbudowana)

MariaDB

- Występuje jako dystrybucja pod MS Windows np. z pakietem **XAMPP** oraz Linux.
- W wiele dystrybucji Linux została już wbudowana lub jest bazą domyślną.
- Obsługa bazy z poziomu konsoli jest podobna


Administrator: C:\Windows\system32\cmd.exe - mysql
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Users\jdugan>set path=C:\wamp\bin\mysql\mysql5.6.12\bin
C:\Users\jdugan>mysql
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 337
Server version: 5.6.12-log MySQL Community Server (GPL)
Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql>
kris@Slamet: ~/Pictures
File Edit Tabs Help
kris@Slamet:~/Pictures$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 53
Server version: 5.5.37-0ubuntu0.14.04.1 (Ubuntu)
Copyright (c) 2000, 2014, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> show global variables like 'have_%ssl';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| have_openssl | YES |
| have_ssl | YES |
+---------------+-------+
2 rows in set (0.00 sec)
mysql>

Logowanie do mysql

Do administracji bazami MySQL najczęściej używa się domyślnie tworzonego konta o nazwie **root**, o nieograniczonych możliwościach.

Czasami, ze względów bezpieczeństwa, tworzy się osobne konto o tych samych uprawnieniach jednak z inną nazwą użytkownika.

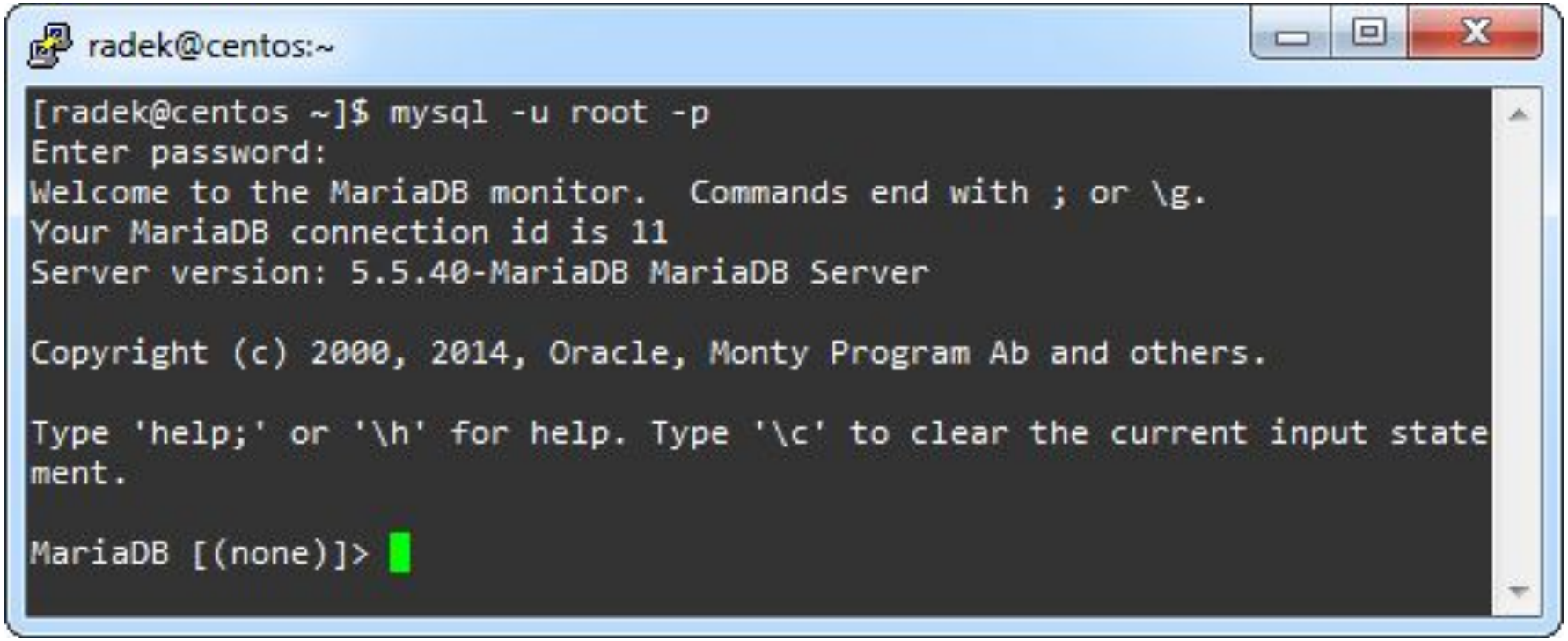
Każda instalowana na serwerze aplikacja, która korzysta z baz danych powinna czynić to za pomocą dedykowanego użytkownika z prawami dostępu wyłącznie do konkretnej bazy.

Należy unikać sytuacji gdy ten sam użytkownik, a tym bardziej root, *obsługuje* różne aplikacje.

Do poprawnego i spójnego działania bazy danych niezbędne jest umieszczenie w każdej tabeli

A. kluczy PRIMARY KEY i FOREIGN KEY
B. klucza FOREIGN KEY z wartością NOT NULL
C. klucza obcego z wartością NOT NULL i UNIQUE
D. **klucza PRIMARY KEY z wartością NOT NULL i UNIQUE**

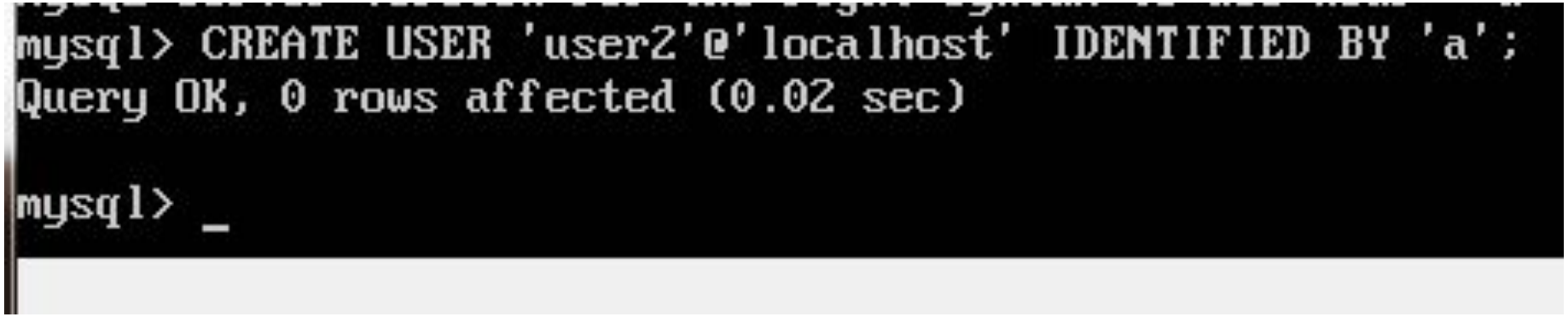
Użytkownik w MySql



Komenda **mysql -u root -p**, po której system poprosi o hasło, które należy podać. Nie podajemy hasła po samym parametrze –p.

Tworzenie użytkownika i nadawanie mu wszystkich praw

$ CREATE USER 'nowyUzytkownik'@'localhost'
IDENTIFIED BY ‘haslo';



Tworzymy użytkownika: **user2** na naszym lokalnym komputerze z hasłem: **a**

GRANT ALL PRIVILEGES ON *.* TO 'nowyUzytkownik'@'localhost';

– nadanie wszystkich praw na wszystkich bazach dla użytkownika **nowyUzytkownik@localhost**.

```
mysql> GRANT ALL PRIVILEGES on *.* TO 'user2'@'localhost';
Query OK, 0 rows affected (0.00 sec)

mysql>
```

U nas użytkownik to **user2**

GRANT ALL PRIVILEGES ON przykladowaBaza.* TO 'nowyUzytkownik'@'localhost';

– nadanie wszystkich praw dla **nowyUzytkownik@localhost**, ale tylko dla bazy **przykladowaBaza**.

Polecenie w języku SQL **GRANT ALL PRIVILEGES ON klienci TO pracownik**:

A. nadaje uprawnienie grupie *klienci* do tabeli *pracownik*

B. odbiera wszystkie uprawnienia *pracownikowi* do tabeli *klienci*

C. skopiuje uprawnienia z grupy *klienci* na użytkownika *pracownik*

D. nadaje wszystkie uprawnienia do tabeli *klienci* użytkownikowi *pracownik*

W serwerze MySQL nadanie **roli o nazwie DBManager** przyznaje użytkownikowi prawa umożliwiające

A. monitorowanie serwera

B. wszelkie operacje na bazach danych serwera

C. tworzenie użytkowników serwera i ustawianie im haseł

D. wszystkie operacje na bazach danych i użytkownikach serwera

Polecenie **REVOKE SELECT ON nazwa1 FROM nazwa2** w języku SQL umożliwia

A. nadanie uprawnień z użyciem zdefiniowanego schematu

B. odbieranie uprawnień użytkownikowi

C. usuwanie użytkownika z bazy

D. nadawanie praw do tabeli

- W języku SQL przywilej SELECT polecenia **GRANT** pozwala użytkownikowi baz danych na

A. odczyt danych z tabeli

B. tworzenie tabeli

C. usunięcie danych z tabeli

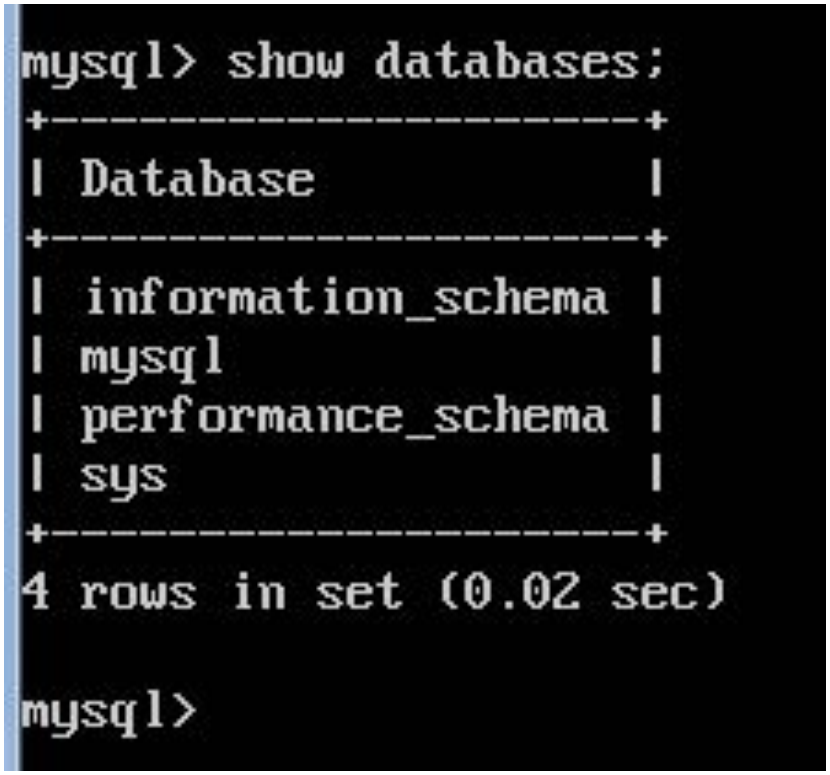
D. modyfikowanie danych w tabeli

Koniec pracy z dodaniem userów

Na sam koniec trzeba przeładować uprawnienia poleceniem **FLUSH PRIVILEGES;** , zaś połączenie z bazą danych zamyka się poleceniem **quit**.

Czy baza istnieje

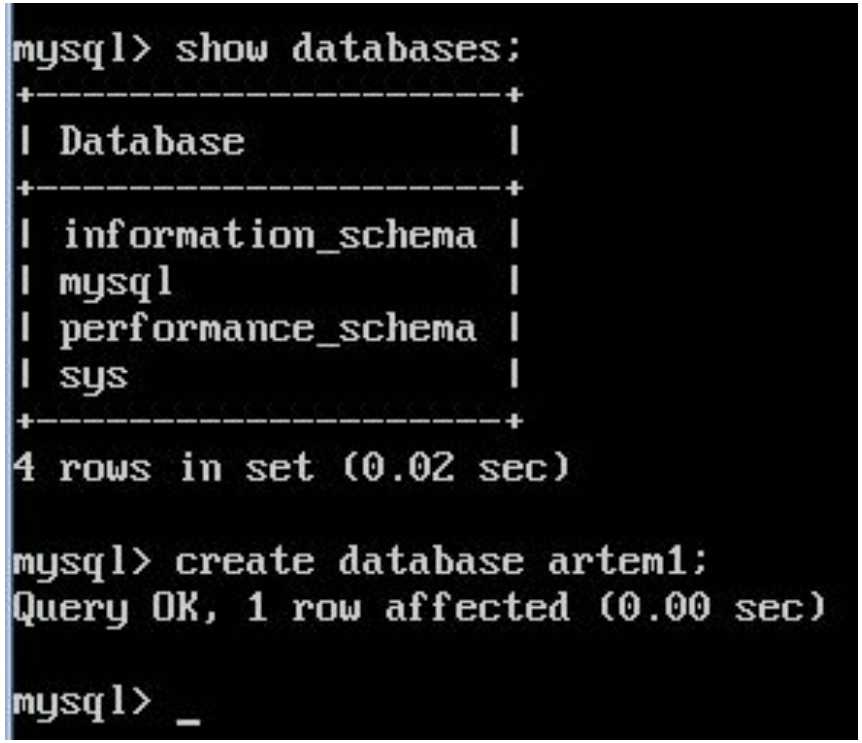
Wyświetlenie istniejących baz danych – polecenie $mysql>**show databases;**



Tworzenie BD

Tworzenie nowej bazy danych:

create database [nazwa_bazy];



- Wskaż PRAWDZIWE stwierdzenie dla polecenia: CREATE TABLE **IF NOT EXISTS** ADRES(ulica VARCHAR(70) CHARACTER SET utf8);

A. Rekordem tabeli nie może być 3 MAJA

B. Klauzula CHARACTER SET utf8 jest obowiązkowa

C. Do tabeli nie można wprowadzać ulic zawierających w nazwie polskie znaki

D. IF NOT EXISTS stosuje się opcjonalnie, aby upewnić się, że brak w bazie danych takiej tabeli

Przejście do tabeli

Przejście (włączenie) do wybranej bazy danych:

use [nazwa bazy];

```
mysql> use artem1;
Database changed
mysql> _
```

Tworzenie nowej tabeli: **create table [nazwa_tabeli]**

```
mysql> create table cars;
ERROR 1113 (42000): A table must have at least 1 column
mysql> _
```

Tabela bez kolumny! Tak się nie da.

Polecenie **DBCC CHECKDB("sklepAGD", Repair_fast)** w MS SQL Server

A. sprawdzi spójność określonej tabeli

B. sprawdzi spójność bazy danych i naprawi uszkodzone indeksy

C. sprawdzi spójność bazy danych i wykona kopię bezpieczeństwa

D. sprawdzi spójność określonej tabeli i naprawi uszkodzone rekordy

- Aby przywrócić bazę danych MS SQL z kopii bezpieczeństwa, należy zastosować polecenie

A. DBCC CHECKDB

B. SAVE DATABASE

C. **RESTORE DATABASE**

D. REBACKUP DATABASE

Które z poleceń naprawi uszkodzoną tabelę w języku SQL?

A. REGENERATE TABLE tbl_name

B. **REPAIR TABLE tblname**

C. OPTIMIZE TABLE tbl_name

D. ANALYZE TABLE tbl_name

REPAIR TABLE tblname

```
mysql> repair table artem1;
+---------------+--------+----------+-----------------------------------------+
| Table         | Op     | Msg_type | Msg_text                                |
+---------------+--------+----------+-----------------------------------------+
| artem1.artem1 | repair | Error    | Table 'artem1.artem1' doesn't exist |
| artem1.artem1 | repair | status   | Operation failed                        |
+---------------+--------+----------+-----------------------------------------+
2 rows in set (0.00 sec)

mysql> _
```

Baza danych MySQL uległa uszkodzeniu. Które z działań NIE pomoże przy jej naprawie?

A. Wykonanie replikacji bazy danych

B. Próba naprawy poleceniem REPAIR

C. Odtworzenie bazy z kopii bezpieczeństwa

D. Stworzenie nowej bazy i przeniesienie do niej tabel

TYP	Opis
`VARCHAR(200)`	Typ tekstowy, w nawiasach podajmy liczbę znaków jakie maksymalnie będzie można zapisać w danej kolumnie (do 8 000)
`INT`	Liczba całkowita (od -2^{31} do $2^{31}-1$)
`FLOAT(2)`	Liczby rzeczywiste z określoną w nawiasach dokładnością (1-24 - 7 cyfr, 25-53 - 15 cyfr)
`DATETIME`	Data i czas (Format: `YYYY-MM-DD HH:mm:SS`, `Y` - Rok, `M` - Miesiąc, `D` - Dzień, `H` - Godzina, `m` - Minuta, `S` - sekunda)
`CHAR`	Łańcuch znaków o długości do 8000 znaków
`TEXT`	Typ tekstowy (maksymalna liczba znaków: 2^{31} - 2 147 483 647)
`MONEY`	Typ danych służący do przechowywania wartości pieniężnych (dokładność do jednej dziesięciotysięcznej jednostki, przechowuje liczby z zakresu: od -2^{63} do $2^{63}-1$

```
mysql> create table cars (id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    -> marka VARCHAR(30),
    -> model varchar(30),
    -> uzyw char(1),
    -> data DATE);
Query OK, 0 rows affected (0.03 sec)

mysql> _
```

Każdy wiersz ciągnę do przecinka i ENTER przechodzę do następnego wiersza itd..

```
mysql> show tables;
+------------------+
| Tables_in_artem1 |
+------------------+
| cars             |
+------------------+
1 row in set (0.00 sec)

mysql>
```

Sprawdzam co też utworzyłem.

mysql> DESCRIBE nazwa_tab ;

```
mysql> describe cars;
+--------+-------------+------+-----+---------+----------------+
| Field  | Type        | Null | Key | Default | Extra          |
+--------+-------------+------+-----+---------+----------------+
| id     | int(11)     | NO   | PRI | NULL    | auto_increment |
| marka  | varchar(30) | YES  |     | NULL    |                |
| model  | varchar(30) | YES  |     | NULL    |                |
| uzyw   | char(1)     | YES  |     | NULL    |                |
| data   | date        | YES  |     | NULL    |                |
+--------+-------------+------+-----+---------+----------------+
5 rows in set (0.01 sec)

mysql>
```

Insert

```
mysql> insert into `cars` (`id`,`marka`,`model`,`uzyw`,`data`) values (NULL, "Ford", "Mondeo", "T",
'2014');
ERROR 1292 (22007): Incorrect date value: '2014' for column 'data' at row 1
mysql> _
```

Tradycyjny problem dotyczący dat w systemach baz danych. By go eliminować stosujemy maski wprowadzania.

```
mysql> insert into `cars` (`id`,`marka`,`model`,`uzyw`,`data`) values (NULL, "Ford", "Mondeo", "T",
'2014-12-31');
Query OK, 1 row affected (0.00 sec)

mysql> _
```

Pod strzałkami góra – dół mamy powtarzanie komend, dodanie kolejnych rekordów będzie szybsze.

W języku SQL polecenie **INSERT INTO**:

A. dodaje tabelę

B. dodaje pola do tabeli

C. wprowadza dane do tabeli

D. aktualizuje rekordy określoną wartością

```
mysql> insert into `cars` (`id`,`marka`,`model`,`uzyw`,`data`) values (NULL, "Fiat", "Punto", "N", '
2010-08-15');
Query OK, 1 row affected (0.02 sec)

mysql> insert into `cars` (`id`,`marka`,`model`,`uzyw`,`data`) values (NULL, "Mercedes", "200D", "N"
, '2000-
Query OK, 1 row affected (0.01 sec)

mysql> insert into `cars` (`id`,`marka`,`model`,`uzyw`,`data`) values (NULL, "Tesla", "X", "N", '201
8-01-25');
Query OK, 1 row affected (0.01 sec)

mysql>
```

Aby policzyć wszystkie wiersze tabeli Koty należy użyć polecenia:

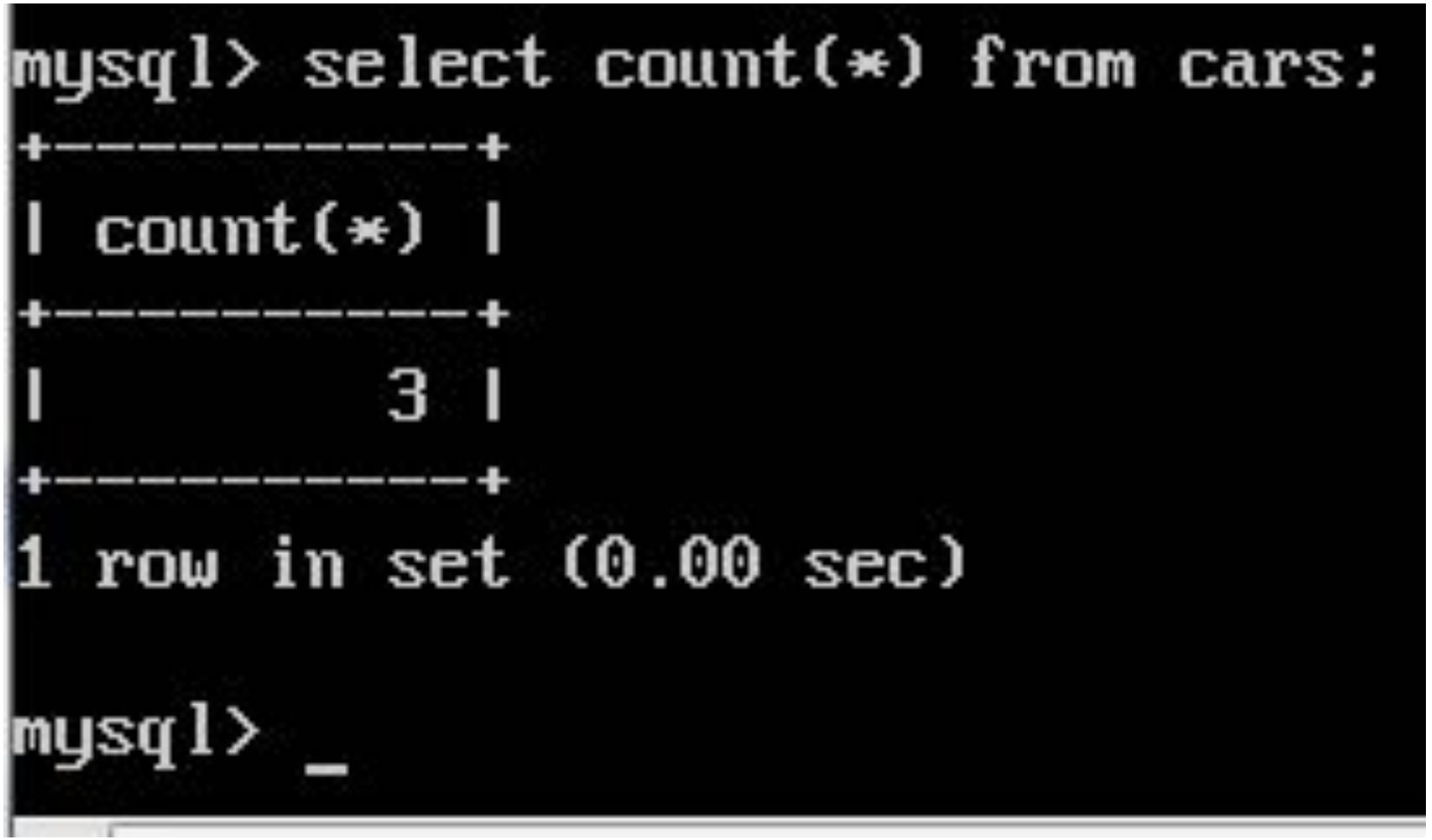
A. SELECT COUNT(*) FROM Koty

B. SELECT ROWNUM() FROM Koty

C. SELECT COUNT(Koty) AS ROWNUM

D. SELECT COUNT(ROWNUM) FROM Koty

SELECT COUNT(*) FROM cars;



mysql> SELECT * FROM cars;

```
mysql> select * from cars;
+----+----------+--------+------+-------------+
| id | marka    | model  | uzyw | data        |
+----+----------+--------+------+-------------+
|  1 | Ford     | Mondeo | T    | 2014-12-31 |
|  2 | Fiat     | Punto  | N    | 2010-08-15 |
|  3 | Mercedes | 200D   | N    | 2000-06-20 |
|  4 | Tesla    | X      | N    | 2018-01-25 |
+----+----------+--------+------+-------------+
4 rows in set (0.01 sec)

mysql> _
```

Jak nazywa się podzbiór strukturalnego języka zapytań, związany z formułowaniem zapytań do bazy danych za pomocą polecenia SELECT?

A. SQL DML (ang. Data Manipulation Language)

B. SQL DDL (ang. Data Definition Language)

C. SQL DCL (ang. Data Control Language)

D. SQL DQL (ang. Data Query Language)

Baza danych zawiera tabelę uczniowie z polami: imie, nazwisko, klasa. Aby odnaleźć imiona i nazwiska tych uczniów, których **nazwiska rozpoczynają się literą M**, należy zastosować polecenie SQL:

A. SELECT nazwisko, imie FROM uczniowie WHERE nazwisko IN "M%";

B. SELECT nazwisko, imie FROM uczniowie WHERE nazwisko LIKE "M%";

C. SELECT nazwisko, imie FROM uczniowie ORDER BY nazwisko = "M%";

D. SELECT nazwisko, imie FROM uczniowie ORDER BY nazwisko IN "M%";

SELECT marka, model FROM cars WHERE marka LIKE "M%";

```
mysql> select marka, model from cars where marka like "M%";
+----------+-------+
| marka    | model |
+----------+-------+
| Mercedes | 200D  |
+----------+-------+
1 row in set (0.00 sec)

mysql> _
```

```
mysql> select marka, model, data from cars where marka like "M%";
+----------+-------+------------+
| marka    | model | data       |
+----------+-------+------------+
| Mercedes | 200D  | 2000-06-20 |
+----------+-------+------------+
1 row in set (0.00 sec)
```

Które z poleceń umożliwia **dodanie kolumny** zadaniekompletne do tabeli zadania?

A. ALTER TABLE zadania ADD COLUMN zadaniekompletne int

B. ADD COLUMN zadaniekompletne WITH zadania

C. CREATEINDEX zadania ADD COLUMN zadaniekompletne int

D. INSERT INTO zadania VALUES zadaniakompletne

ALTER TABLE cars ADD COLUMN URL int;

```
mysql> alter table cars add column URL int;
Query OK, 0 rows affected (0.10 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> select * from cars;
+----+----------+-------+------+------------+-------+
| id | marka    | model | uzyw | data       | URL   |
+----+----------+-------+------+------------+-------+
|  2 | Fiat     | Punto | N    | 2010-08-15 | NULL  |
|  3 | Mercedes | 200D  | N    | 2000-06-20 | NULL  |
|  4 | Tesla    | X     | N    | 2018-01-25 | NULL  |
+----+----------+-------+------+------------+-------+
3 rows in set (0.00 sec)
```

Kasowanie kolumny URL – błędny typ danych

```
mysql> alter table cars drop URL;
Query OK, 0 rows affected (0.11 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> select * from cars;
+----+-----------+--------+------+--------------+
| id | marka     | model | uzyw | data         |
+----+-----------+--------+------+--------------+
|  2 | Fiat      | Punto | N    | 2010-08-15 |
|  3 | Mercedes | 200D  | N    | 2000-06-20 |
|  4 | Tesla     | X     | N    | 2018-01-25 |
+----+-----------+--------+------+--------------+
3 rows in set (0.00 sec)
```

Dodanie prawidłowego typu kolumny

```
mysql> alter table cars add column URL varchar(30);
Query OK, 0 rows affected (0.09 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
```

Dodanie nowej kolumny

mysql> ALTER TABLE spotkanie ADD email VARCHAR(40) ;

```
mysql> alter table cars add e-mail VARCHAR(40);
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your
MySQL server version for the right syntax to use near '-mail VARCHAR(40)' at line 1
mysql> alter table cars add e-mail VARCHAR(40);
```

```
mysql> alter table cars add email VARCHAR(40);
Query OK, 0 rows affected (0.10 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> _
```

Wskaż polecenie SQL **dodające pole** miesiacSiewu do istniejącej tabeli rosliny

A. UPDATE rosliny ADD miesiacSiewu int

B. CREATE TABLE rosliny {miesiacSiewu int}

C. ALTER TABLE rosliny ADD miesiacSiewu int

D. INSERT INTO rosliny VALUES (miesiacSiewu int)

```
mysql> select * from cars;
+----+-----------+--------+------+-------------+--------+
| id | marka     | model  | uzyw | data        | email  |
+----+-----------+--------+------+-------------+--------+
|  1 | Ford      | Mondeo | T    | 2014-12-31 | NULL   |
|  2 | Fiat      | Punto  | N    | 2010-08-15 | NULL   |
|  3 | Mercedes  | 200D   | N    | 2000-06-20 | NULL   |
|  4 | Tesla     | X      | N    | 2018-01-25 | NULL   |
+----+-----------+--------+------+-------------+--------+
4 rows in set (0.00 sec)

mysql>
```

Jak uzupełnić poszczególne komórki w naszej tabeli?

Kiedy już posiadamy tabelę „**cars**", możemy w niej wprowadzać różnie zmiany.
Dla przykładu, właściciel Mercedesa ma email a@a.pl

```
mysql> update `cars` set `email`='a@a.pl' where `cars`.`marka`='Mercedes';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> _
```

```
mysql> update `cars` set `email`='z@z.pl' where `cars`.`id`='1';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from cars;
+----+----------+--------+------+------------+---------+
| id | marka    | model  | uzyw | data       | email   |
+----+----------+--------+------+------------+---------+
|  1 | Ford     | Mondeo | T    | 2014-12-31 | z@z.pl  |
|  2 | Fiat     | Punto  | N    | 2010-08-15 | NULL    |
|  3 | Mercedes | 200D   | N    | 2000-06-20 | a@a.pl  |
|  4 | Tesla    | X      | N    | 2018-01-25 | NULL    |
+----+----------+--------+------+------------+---------+
4 rows in set (0.00 sec)
```

W języku SQL, wykorzystywanym przez bazę danych MySQL w tabeli samochody, aby nadać wartość równą 0 dla kolumny przebieg, należy posłużyć się kwerendą

A. UPDATE samochody SET przebieg = 0;

B. UPDATE przebieg SET 0 FROM samochody;

C. UPDATE przebieg SET 0 TABLE samochody;

D. UPDATE samochody SET przebieg VALUE 0;

```
mysql> alter table cars add column przebieg int;
Query OK, 0 rows affected (0.09 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
```

```
mysql> update cars set przebieg = 0;
Query OK, 3 rows affected (0.03 sec)
Rows matched: 3  Changed: 3  Warnings: 0

mysql> select * from cars;
+----+----------+-------+------+------------+------+----------+
| id | marka    | model | uzyw | data       | URL  | przebieg |
+----+----------+-------+------+------------+------+----------+
|  2 | Fiat     | Punto | N    | 2010-08-15 | NULL |        0 |
|  3 | Mercedes | 200D  | N    | 2000-06-20 | NULL |        0 |
|  4 | Tesla    | X     | N    | 2018-01-25 | NULL |        0 |
+----+----------+-------+------+------------+------+----------+
3 rows in set (0.00 sec)
```

Aktualizacja przebiegów dla marek samochodów

```
mysql> update cars set przebieg = 40000 where marka="Fiat";
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from cars;
+----+----------+-------+------+------------+------+----------+
| id | marka    | model | uzyw | data       | URL  | przebieg |
+----+----------+-------+------+------------+------+----------+
|  2 | Fiat     | Punto | N    | 2010-08-15 | NULL |    40000 |
|  3 | Mercedes | 200D  | N    | 2000-06-20 | NULL |        0 |
|  4 | Tesla    | X     | N    | 2018-01-25 | NULL |        0 |
+----+----------+-------+------+------------+------+----------+
3 rows in set (0.00 sec)
```

Baza danych 6-letniej szkoły podstawowej zawiera tabelę *szkola* z polami: *imie, nazwisko, klasa*. Wszyscy uczniowie klas 1-5 zdali do następnej klasy. Aby zwiększyć wartość w polu klasa o 1 należy użyć polecenia

A. SELECT szkola FROM klasa=klasa+1 WHERE klasa >=1 AND klasa <=5;

B. SELECT nazwisko, imie FROM klasa=klasa+1 WHERE klasa>1 OR klasa <5;

C. UPDATE szkola SET klasa=klasa+1 WHERE klasa>=1 AND klasa <=5;

D. UPDATE nazwisko, imie SET klasa=klasa+1 WHERE klasa>1 OR klasa<5;

Usuwamy kolumnę

- **ALTER TABLE spotkanie DROP email ;**

```
mysql> alter table cars DROP email;
Query OK, 0 rows affected (0.09 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> _
```

Polecenie w języku SQL **ALTER TABLE USA**... ma za zadanie

A. usunięcie tabeli USA

B. modyfikację tabeli USA

C. nadpisanie starej tabeli USA

D. utworzenie nowej tabeli USA

W języku SQL w wyniku wykonania zapytania **ALTER TABLE osoba DROP COLUMN grupa**; zostanie:

A. dodana kolumna grupa

B. usunięta kolumna grupa

C. zmieniona nazwa tabeli na grupa

D. zmieniona nazwa kolumny na grupa

Kasowanie rekordu

DELETE from [table name] where [column name]=[field text] ;

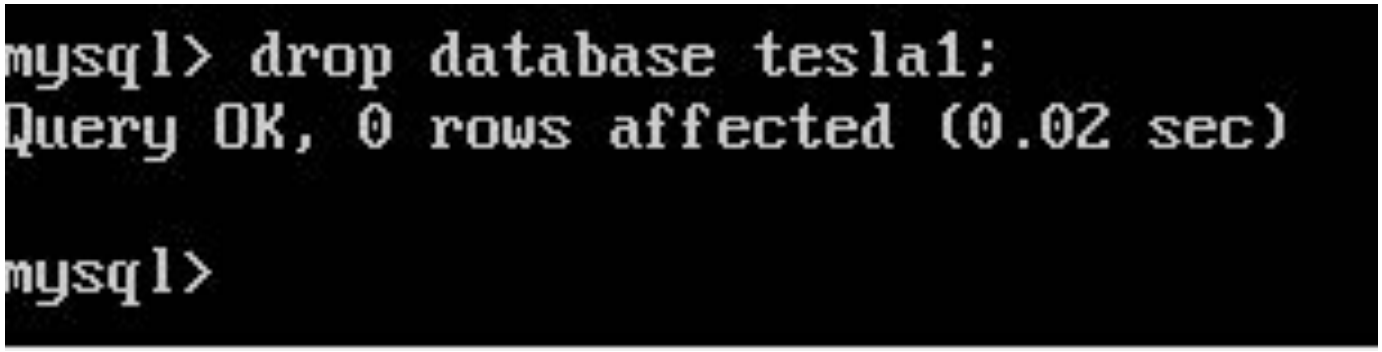
```
mysql> DELETE FROM cars WHERE id='1';
Query OK, 1 row affected (0.01 sec)

mysql> select * from cars;
+----+----------+-------+------+------------+
| id | marka    | model | uzyw | data       |
+----+----------+-------+------+------------+
|  2 | Fiat     | Punto | N    | 2010-08-15 |
|  3 | Mercedes | 200D  | N    | 2000-06-20 |
|  4 | Tesla    | X     | N    | 2018-01-25 |
+----+----------+-------+------+------------+
3 rows in set (0.00 sec)

mysql> _
```

usunięcie bazy danych wystarczy wpisać:

- **DROP DATABASE tesla1;**



Instrukcja **DROP** języka SQL ma za zadanie:

A. usunąć istniejący obiekt

B. zmienić parametry obiektu

C. zaktualizować dane obiektu

D. dodać nowy obiekt

Sortowanie bazy danych

>SELECT nazwa_pola FROM nazwa_tabeli

ORDER BY nazwa_pola [rodzaj_sortowania];

Rodzaj sortowania nie jest obowiązkowy.
Dostępne są dwa rodzaje sortowania:

- **ASC** - sortowanie rosnąco, domyślny sposób sortowania
- **DESC** - sortowanie malejąco

Zdefiniowano bazę danych z tabelą podzespoły o polach: model, producent, typ, cena. Aby wyświetlić wszystkie modele pamięci RAM firmy Kingston w kolejności od najtańszej do najdroższej, należy posłużyć się kwerendą:

A. SELECT model FROM podzespoly WHERE typ="RAM" AND producent="Kingston" ORDER BY cena ASC;

B. SELECT model FROM podzespoly WHERE typ="RAM" AND producent="Kingston" ORDER BY cena DESC;

C. SELECT model FROM podzespoly WHERE typ="RAM" OR producent="Kingston" ORDER BY cena DESC;

D. SELECT model FROM producent WHERE typ="RAM" OR producent="Kingston" ORDER BY podzespoly ASC;

```
mysql> select * from `cars` order by `cars`.`id` ASC;
+----+----------+-------+------+------------+
| id | marka    | model | uzyw | data       |
+----+----------+-------+------+------------+
|  2 | Fiat     | Punto | N    | 2010-08-15 |
|  3 | Mercedes | 200D  | N    | 2000-06-20 |
|  4 | Tesla    | X     | N    | 2018-01-25 |
+----+----------+-------+------+------------+
3 rows in set (0.00 sec)

mysql> select * from `cars` order by `cars`.`id` desc;
+----+----------+-------+------+------------+
| id | marka    | model | uzyw | data       |
+----+----------+-------+------+------------+
|  4 | Tesla    | X     | N    | 2018-01-25 |
|  3 | Mercedes | 200D  | N    | 2000-06-20 |
|  2 | Fiat     | Punto | N    | 2010-08-15 |
+----+----------+-------+------+------------+
3 rows in set (0.00 sec)

mysql>
```

W bazie danych, w celu uporządkowania listy uczniów według roku urodzenia, należy użyć polecenia

A. SELECT imie,nazwisko,klasa from uczniowie group by rok_urodzenia

B. SELECT imie,nazwisko,klasa from uczniowie order by rok_urodzenia

C. SELECT imie,nazwisko,klasa from uczniowie order by nazwisko

D. SELECT imie,nazwisko,klasa from uczniowie where rok_urodzenia = 1994

- W bazie danych sklepu spożywczego pod koniec dnia jest tworzony raport wyświetlający te produkty wraz z ich dostawcami, dla których stan magazynowy jest mniejszy niż 10 sztuk. Do zdefiniowania tego raportu posłużono się kwerendą

- **A. SELECT**
- **B.** UPDATE
- **C.** INSERT INTO
- **D.** CHECK TABLE

Polecenie SQL o treści: **UPDATE artykuly SET cena = cena * 0.7 WHERE kod = 2**; oznacza:

- **A.** w tabeli artykuły obniża wartość każdego pola cena o 30% dla wszystkich artykułów
- **B. w tabeli artykuly obniża wartość każdego pola cena dla którego pole kod jest równe 2**
- **C.** wprowadzenie w tabeli artykuly nowych pól cena i kod
- **D.** wprowadzenie w tabeli artykuly pola o nazwie cena ze znacznikiem kod

Dana jest tabela o nazwie przedmioty z polami: **ocena i uczenID**. Aby policzyć średnią ocen ucznia o **ID równym 7**, należy posłużyć się zapytaniem:

A. AVG SELECT ocena FROM przedmioty WHERE uczenID = 7;

B. **SELECT AVG(ocena) FROM przedmioty WHERE uczenID = 7;**

C. COUNT SELECT ocena FROM przedmioty WHERE uczenID = 7;

D. SELECT COUNT(ocena) FROM przedmioty WHERE uczenID = 7;

Którą klauzulę powinno się zastosować w poleceniu **CREATE TABLE** języka SQL, aby dane pole rekordu **nie było puste**?

A. NULL

B. CHECK

C. DEFAULT

D. **NOT NULL**

Dana jest tabela psy o polach: **imie, rasa, telefon_wlasciciela, rok_szczepienia**. Aby wyszukać telefony właścicieli, których psy były szczepione przed 2015 rokiem, należy użyć polecenia SQL:

A. SELECT psy FROM rok_szczepienia < 2015

B. SELECT imie, rasa FROM psy WHERE rok_szczepienia > 2015

C. SELECT telefon_wlasciciela FROM psy WHERE rok_szczepienia < 2015

D. SELECT telefon_wlasciciela FROM psy WHERE rok_szczepienia > 2015

Jakie sa nazwy typowych poleceń języka zapytań SQL, związane z wykonywaniem operacji na danych **SQL DML** (np.: umieszczanie danych w bazie, kasowanie dokonywanie zmian w danych)?

A. SELECT, SELECT INTO

B. ALTER, CREATE, DROP

C. DENY, GRANT, REVOKE

D. **DELETE, INSERT, UPDATE**

Dodanie kolumny

```
mysql> alter table cars add column podzespoly varchar(30);
Query OK, 0 rows affected (0.08 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

W tabeli **podzespoly** należy zmienić wartość pola **URL** na '**toshiba.pl**' dla wszystkich rekordów, gdzie pole producent to **TOSHIBA**. W języku SQL modyfikacja będzie miała postać

A. UPDATE podzespoly SET URL='toshiba.pl';

B. UPDATE producent='TOSHIBA' SET URL='toshiba.pl';

C. UPDATE podzespoly.producent='TOSHIBA' SET URL='toshiba.pl';

D. UPDATE podzespoly SET URL='toshiba.pl' WHERE producent='TOSHIBA';

Dodanie do pola URL rekordów toschiba.pl

```
mysql> select * from cars;
+----+----------+-------+------+------------+-------------+----------+------------+
| id | marka    | model | uzyw | data       | URL         | przebieg | podzespoly |
+----+----------+-------+------+------------+-------------+----------+------------+
|  2 | Fiat     | Punto | N    | 2010-08-15 | toschiba.pl |    40000 | NULL       |
|  3 | Mercedes | 200D  | N    | 2000-06-20 | toschiba.pl |        0 | NULL       |
|  4 | Tesla    | X     | N    | 2018-01-25 | toschiba.pl |        0 | NULL       |
+----+----------+-------+------+------------+-------------+----------+------------+
3 rows in set (0.00 sec)

mysql> update cars set URL='toschiba.pl';
```

Baza danych 6-letniej szkoły podstawowej zawiera tabelę ***szkola*** z polami: *imie, nazwisko, klasa*. Wszyscy uczniowie klas 1-5 zdali do następnej klasy. Aby zwiększyć wartość w polu klasa o 1 należy użyć polecenia

A. SELECT szkola FROM klasa=klasa+1 WHERE klasa >=1 AND klasa <=5;

B. SELECT nazwisko, imie FROM klasa=klasa+1 WHERE klasa>1 OR klasa <5;

C. UPDATE szkola SET klasa=klasa+1 WHERE klasa>=1 AND klasa <=5;

D. UPDATE nazwisko, imie SET klasa=klasa+1 WHERE klasa>1 OR klasa<5;

Kod: **SELECT imie, pesel, wiek FROM dane WHERE wiek IN (18,30) spowoduje wybranie**:

A. imion, nazwisk i numerów PESEL osób w wieku poniżej 18 lat

B. imion, numerów PESEL i wieku osób z przedziału od 18 do 30 lat

C. imion, numerów PESEL i wieku osób posiadających powyżej 30 lat

D. imion, numerów PESEL i wieku osób w wieku równym 18 lub 30 lat

- W bazie danych MYSQL dana jest tabela programów komputerowych o polach: **nazwa, producent, rokWydania**. Aby kwerenda SELECT zwróciła wszystkie nazwy producentów tak, by nazwy te nie powtarzały się, należy zapisać:

A. SELECT UNIQUE producent FROM programy;

B. SELECT DISTINCT producent FROM programy;

C. SELECT producent FROM programy WHERE UNIQUE;

D. SELECT producent FROM programy WHERE producent NOT DUPLICATE;

Funkcja agregująca **MIN** języka SQL ma za zadanie policzyć

A. Liczbę wierszy zwróconych kwerendą

B. **Wartość minimalną kolumny zwróconej kwerendą**

C. długość znaków w zwróconych kwerendą rekordach

D. Średnią wartości różnych pól rekordu zwróconego zapytaniem

W języku SQL wykorzystywanym przez bazę danych MySQL atrybut **UNIQUE** polecenia **CREATE TABLE**

A. Wymusza unikatowe nazwy pól tabeli

B. Blokuje możliwość wpisania wartości NULL

C. Jest stosowany tylko w przypadku pól liczbowych

D. Jest stosowany, jeśli wartość w kolumnie nie mogą się powtarzać

- Dana jest tabela o nazwie wycieczki z polami: **nazwa, cena, miejsca** (jako liczba wolnych miejsc). Aby dla dowolnego zbioru danych tabeli wyświetlić jedynie nazwy tych wycieczek, dla których cena jest niższa niż **2000 zł** i mają przynajmniej cztery wolne miejsca, należy posłużyć się zapytaniem:

- **A. SELECT nazwa FROM wycieczki WHERE cena < 2000 AND miejsca > 3;**
- **B.** SELECT nazwa FROM wycieczki WHERE cena < 2000 OR miejsca > 4;
- **C.** SELECT * FROM wycieczki WHERE cena < 2000 AND miejsca > 4;
- **D.** SELECT * FROM wycieczki WHERE cena < 2000 OR miejsca > 3;

Które polecenie wydane z konsoli systemowej dokona przywrócenia bazy danych?

A. mysqldump -u root -p baza > kopia.sql

B. mysqldump -u root -p baza < kopia.sql

C. **mysql -u root -p baza < kopia.sql**

D. mysql -u root -p baza > kopia.sql

Dana jest tabela programiści o polach: **id**, **nick**, **ilosc_kodu**, **ocena**. Pole **ilosc_kodu** zawiera liczbę linii kodu napisanych przez programistę w danym miesiącu. Aby policzyć **sumę linii kodu**, który napisali wszyscy programiści, należy użyć polecenia

A. SELECT SUM(ocena) FROM ilosc_kodu;

B. SELECT SUM(ilosc_kodu) FROM programisci;

C. SELECT COUNT(programisci) FROM ilosc_kodu;

D. SELECT MAX(ilosc_kodu) FROM programisci

Za pomocą polecenia **BACKUP LOG** w MS SQL Server można:

A. wykonać pełną kopię bezpieczeństwa

B. zalogować sie do kopii bezpieczeństwa

C. wykonać kopię bezpieczeństwa dziennika transakcyjnego

D. przeczytać komunikaty wygenerowane podczas tworzenia kopii

W bazie danych hurtowni zdefiniowano tabelę **sprzedaz** o polach: id, kontrahent, **grupa_cenowa**, **obrot**. Aby wyszukać wyłącznie kontrahentów z drugiej grupy cenowej, których obrót jest większy niż **4000zł**, należy zastosować polecenie:

A. SELECT sprzedaz FROM kontrahent WHERE obrot > 4000;

B. SELECT kontrahent FROM sprzedaz WHERE grupa_cenowa = 2 OR obrot > 4000;

C. SELECT kontrahent FROM sprzedaz WHERE grupa_cenowa = 2 AND obrot > 4000;

D. SELECT sprzedaz FROM kontrahent WHERE grupa_cenowa = 2 AND obrot > 4000;

Które ze stwierdzeń prawidłowo charakteryzuje zdefiniowaną tabelę: CREATE TABLE dane (kolumna INTEGER(3));

A. Tabela o nazwie *dane* posiada trzy kolumny liczb całkowitych

B. Tabela o nazwie *dane* posiada jedną kolumnę liczb całkowitych

C. Tabela posiada jedną kolumnę zawierającą trzy elementowe tablice

D. Kolumny tabeli *dane* nazywają się: *kolumna1*, *kolumna2*, *kolumna3*

Aby podczas tworzenia tabeli utworzyć klucz obcy na wielu kolumnach, należy użyć polecenia

A. CONSTRAINT(nazwisko,imie) FOREIGN KEY REFERENCES osoby (nazwisko, imie)
B. CONSTRAINT(nazwisko,imie) FOREIGN REFERENCES KEY osoby (nazwisko, imie)
C. **CONSTRAINT fk_osoba_uczen FOREIGN KEY (nazwisko, imie) REFERENCES osoby (nazwisko,imie)**
D. CONSTRAINT fk_osoba_uczen FOREIGN KEY ON (nazwisko, imie) REFERENCES osoby (nazwisko,imie)

Tabela filmy zawiera klucz główny **id** oraz klucz obcy **rezyserID**. Tabela **rezyserzy** zawiera klucz główny id. Obydwie tabele połączone są relacją jeden po stronie rezyserzy do wielu po stronie filmy. Aby w kwerendzie **SELECT** połączyć tabele filmy i **rezyserzy**, należy zapisać

A. ... filmy JOIN rezyserzy ON filmy.id = rezyserzy.id ...

B. ... filmy JOIN rezyserzy ON filmy.id = rezyserzy.filmyID ...

C. ... filmy JOIN rezyserzy ON filmy.rezyserID = rezyserzy.id ...

D. ... filmy JOIN rezyserzy ON filmy.rezyserID = rezyserzy.filmyID ...