

Bazy danych



Andrzej Łachwa, UJ, 2015

andrzej.lachwa@uj.edu.pl

7/14

Zależności funkcyjne

Wszystkie atrybuty schematu relacyjnej bazy danych nazwiemy, w celu ułatwienia prezentacji pewnych problemów teoretycznych, relacją uniwersalną schematu.

$$R = \{A_1, A_2, \dots, A_n\}$$

Zależność funkcyjna to wiązanie integralności między atrybutami:

$$X \rightarrow A, \text{ gdzie } X \subset R \text{ i } A \in R,$$

które oznacza, że dla dowolnych krotek t_1 i t_2 należących do relacji $r(R)$: jeżeli $t_1[X] = t_2[X]$ to $t_1[A] = t_2[A]$.

Innymi słowy, wartości atrybutów tworzących zbiór X krotki t relacji $r(R)$ jednoznacznie determinują wartość tej krotki dla atrybutu A .

Zależność funkcyjna jest własnością schematu R i ma być spełniona przez wszystkie relacje $r(R)$ rozpięte na tym schemacie.

Dla uproszczenia zapisów przyjmujemy, że

$$X \rightarrow Y, \text{ gdzie } X \subseteq R, A_i \in R, \{A_1, A_2, \dots, A_k\} = Y$$

oznacza $X \rightarrow A_1, X \rightarrow A_2, \dots, X \rightarrow A_k$.

Dla danego schematu relacji o atrybutach tworzących zbiór X , jeżeli Y jest kluczem kandydującym (każda krotka ma inną niepustą wartość na atrybutach Y), to $Y \rightarrow Z$ dla dowolnego podzbioru Z zbioru atrybutów X .

Uwaga: przyjmujemy, że klucz kandydujący jest kluczem minimalnym, tzn., że nie można z niego usunąć żadnego atrybutu bez utraty własności identyfikowania krotek.

Zależność funkcyjna jest własnością semantyczną(!), a nie własnością formalną. Zależność funkcyjna jest odkrywana przez projektanta. Nie da się jej udowodnić. Można ją tylko uzasadnić właściwościami obszaru analizy.

Podobnie jest z innymi więzami integralności. Określenie dziedziny atrybutu, możliwość przyjmowania wartości pustej i jej interpretacja, określenie kluczy kandydujących i wybranie klucza głównego – to wszystko decyzje projektanta oparte na jego znajomości obszaru analizy.

Projektant określa zależności funkcyjne schematu R , które są semantycznie oczywiste. Niech będzie to zbiór zależności F .

Czy dla wszystkich relacji $r(R)$ rozpiętych na tym schemacie mogą istnieć jakieś inne zależności funkcyjne (nie wskazane przez projektanta)?

Zwykle tak. Można je wydedukować ze zbioru zależności F i pewnych reguł.

Wszystkie zależności zbioru F i wszystkie zależności funkcyjne które można wydedukować ze zbioru F nazywamy domknięciem tego zbioru i oznaczamy F^+ .

Reguły wnioskowania dla zależności funkcyjnych

$\vdash X \rightarrow X$ (zwrotność)

$X \rightarrow Y \vdash XZ \rightarrow YZ$ (powiększanie)

$X \rightarrow Y, Y \rightarrow Z \vdash X \rightarrow Z$ (przechodność)

$X \rightarrow YZ \vdash X \rightarrow Y$ (dekompozycja)

$X \rightarrow Y, X \rightarrow Z \vdash X \rightarrow YZ$ (sumowanie)

$X \rightarrow Y, WY \rightarrow Z \vdash WX \rightarrow Z$ (pseudoprzechodność)

gdzie napis PQ oznacza dla atrybutów $\{P, Q\}$ a dla zbiorów atrybutów $P \cup Q$, oraz \vdash jest znakiem inferencji.

Pierwsze trzy reguły noszą nazwę aksjomatów Armstronga, albo (lepiej) reguł wnioskowania Armstronga.

Mają one tę własność, że można z nich wyprowadzić pozostałe reguły, ale nie można usunąć żadnej z nich, bo stracą tę własność.

Domknięcie zbioru zależności F można zatem uzyskać stosując do tych zależności reguły Armstronga. Robi się to zwykle poprzez domknięcie każdego zbioru X (stojącego po lewej stronie zależności funkcyjnej) względem zbioru zależności F .

Dwa zbiory zależności funkcyjnych E i F są równoważne wtw gdy $E^+ = F^+$.

Algorytm domknięcia zbioru X względem F

```
 $X^+ := X;$   
repeat  
   $oldX^+ := X^+;$   
  for each zależność funkcyjna  $Y \rightarrow Z$  w  $F$  do  
    if  $X^+ \supset Y$  then  $X^+ := X^+ \cup Z;$   
until  $X^+ = oldX^+;$ 
```

O minimalnych zbiorach zależności funkcyjnych zobacz:

[Ramez Elmasri, Shamkant B. Navathe, str. 336]

Statystyki w języku SQL

W różnych produktach SQL spotkamy rozmaite funkcje wbudowane ułatwiające analizy statystyczne. Jeżeli nie zależy nam na przenośności kodu, to lepiej użyć funkcji wbudowanych.

Zaczniemy od najprostszych statystyk (dla bazy World):

```
select c.continent, count(c.continent) as frequency,  
       round(100.0*(count(c.continent))/  
             (select count(*) from country),2) as percent  
from country as c  
group by continent;
```

```
create temporary table statistics as  
  select c.continent, count(c.continent) frequency,  
         round(100.0*(count(c.continent))/  
               (select count(*) from country),2) as percent  
  from country as c  
  group by continent  
  order by percent desc;
```

```
C:\Windows\system32\cmd.exe

mysql> select c.continent, count(c.continent) frequency,
-> round(100.0*(count(c.continent))/(select count(*) from country),2) as percent
-> from country as c group by continent;
+-----+-----+-----+
| continent | frequency | percent |
+-----+-----+-----+
| Asia      | 51        | 21.34    |
| Europe    | 46        | 19.25    |
| North America | 37      | 15.48    |
| Africa    | 58        | 24.27    |
| Oceania   | 28        | 11.72    |
| Antarctica | 5         | 2.09     |
| South America | 14       | 5.86     |
+-----+-----+-----+
7 rows in set (0.00 sec)

mysql>
```

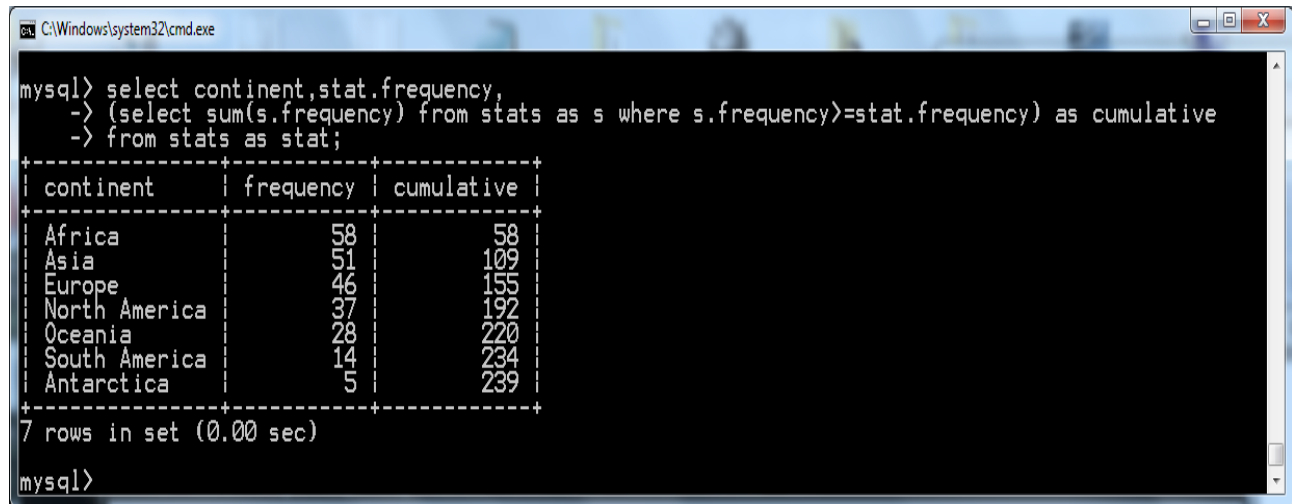
```
C:\Windows\system32\cmd.exe

mysql> create temporary table statistics as
-> select c.continent, count(c.continent) frequency,
-> round(100.0*(count(c.continent))/(select count(*) from country),2) as percent
-> from country as c group by continent order by percent desc;
Query OK, 7 rows affected (0.42 sec)
Records: 7 Duplicates: 0 Warnings: 0

mysql> select * from statistics;
+-----+-----+-----+
| continent | frequency | percent |
+-----+-----+-----+
| Africa    | 58        | 24.27    |
| Asia      | 51        | 21.34    |
| Europe    | 46        | 19.25    |
| North America | 37      | 15.48    |
| Oceania   | 28        | 11.72    |
| South America | 14       | 5.86     |
| Antarctica | 5         | 2.09     |
+-----+-----+-----+
7 rows in set (0.00 sec)

mysql>
```

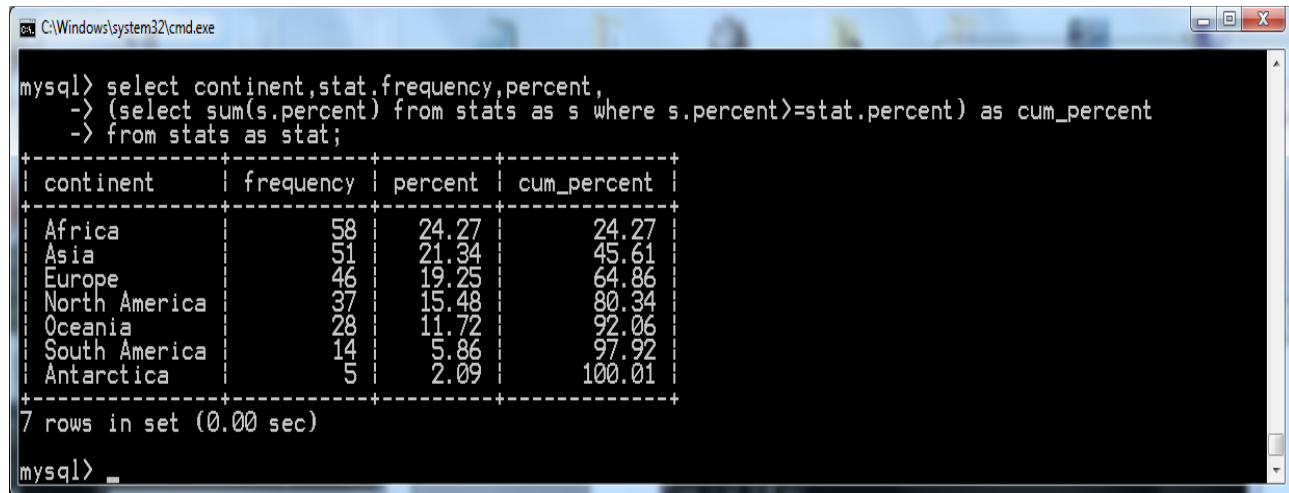
```
select continent,stat.frequency,  
    (select sum(s.frequency) from stats as s  
     where s.frequency>=stat.frequency) as cumulative  
from stats as stat;
```



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". Inside the window, a MySQL query is executed. The query selects the continent, frequency, and a cumulative frequency for each continent. The results are displayed in a table format with dashed lines separating the columns. The table has three columns: continent, frequency, and cumulative. The data rows are: Africa (58, 58), Asia (51, 109), Europe (46, 155), North America (37, 192), Oceania (28, 220), South America (14, 234), and Antarctica (5, 239). Below the table, it says "7 rows in set (0.00 sec)". The prompt "mysql>" is visible at the bottom.

```
C:\Windows\system32\cmd.exe  
mysql> select continent,stat.frequency,  
    -> (select sum(s.frequency) from stats as s where s.frequency>=stat.frequency) as cumulative  
    -> from stats as stat;  
+-----+-----+-----+  
| continent | frequency | cumulative |  
+-----+-----+-----+  
| Africa   | 58        | 58         |  
| Asia     | 51        | 109        |  
| Europe   | 46        | 155        |  
| North America | 37      | 192        |  
| Oceania  | 28        | 220        |  
| South America | 14      | 234        |  
| Antarctica | 5        | 239        |  
+-----+-----+-----+  
7 rows in set (0.00 sec)  
mysql>
```

```
select continent, stat.frequency, percent,  
    (select sum(s.percent) from stats as s  
     where s.percent>=stat.percent) as cum_percent  
from stats as stat;
```



A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window contains a MySQL command prompt session. The user enters a query to select continent, frequency, percent, and cumulative percent from a table named 'stats'. The results are displayed in a table format with 7 rows. The cumulative percent column shows the running total of the percent values for each continent, ordered from highest to lowest frequency.

```
mysql> select continent,stat.frequency,percent,  
-> (select sum(s.percent) from stats as s where s.percent>=stat.percent) as cum_percent  
-> from stats as stat;
```

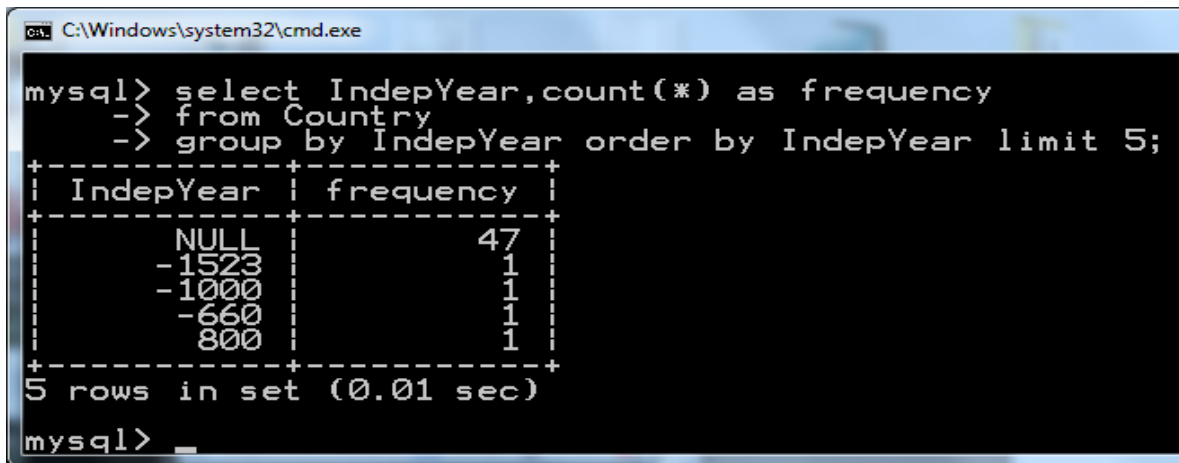
continent	frequency	percent	cum_percent
Africa	58	24.27	24.27
Asia	51	21.34	45.61
Europe	46	19.25	64.86
North America	37	15.48	80.34
Oceania	28	11.72	92.06
South America	14	5.86	97.92
Antarctica	5	2.09	100.01

```
7 rows in set (0.00 sec)  
  
mysql> _
```

Wartość modalna

Wartość modalna to wartość najczęściej pojawiająca się w zbiorze (jeśli są takie dwie wartości, to mówimy że rozkład jest dwumodalny, podobnie trzymodalny etc.).

Wykonajmy to zapytanie w kilku krokach:



```
C:\Windows\system32\cmd.exe

mysql> select IndepYear,count(*) as frequency
-> from Country
-> group by IndepYear order by IndepYear limit 5;
+-----+-----+
| IndepYear | frequency |
+-----+-----+
| NULL      | 47        |
| -1523     | 1         |
| -1000     | 1         |
| -660      | 1         |
| 800       | 1         |
+-----+-----+
5 rows in set (0.01 sec)

mysql> _
```

C:\Windows\system32\cmd.exe

```
mysql> select IndepYear,count(*) as frequency  
-> from Country  
-> group by IndepYear  
-> order by count(*) DESC limit 5;
```

IndepYear	frequency
NULL	47
1960	18
1991	18
1962	7
1975	7

5 rows in set (0.00 sec)

```
mysql> _
```


C:\Windows\system32\cmd.exe

```
mysql> select IndepYear,count(*) as modalfrequency
-> from Country
-> group by IndepYear
-> having count(*)>=
-> all (select count(*) from Country group by IndepYear);
```

IndepYear	modalfrequency
NULL	47

1 row in set (0.00 sec)

```
mysql>
```

C:\Windows\system32\cmd.exe

```
mysql> select IndepYear,count(*) as m_frequency  
-> from Country  
-> where IndepYear is not NULL  
-> group by IndepYear  
-> having count(*)>=  
-> all (select count(*) from Country group by IndepYear);
```

Empty set (0.00 sec)

```
mysql>
```

```
mysql>
```

```
mysql>
```

C:\Windows\system32\cmd.exe

```
mysql>
mysql> select IndepYear, count(*) as m_frequency
-> from Country
-> where IndepYear is not NULL
-> group by IndepYear
-> having count(*) >=
-> all (select count(*) from Country
->       where IndepYear is not NULL group by IndepYear);
```

IndepYear	m_frequency
1960	18
1991	18

2 rows in set (0.00 sec)

```
mysql>
```

Wartość modalna jest słabą statystyką. Na przykład

1. gdy liczymy modalne pobory wśród pracowników, to dokładna wartość tej zmiennej nie jest dobra – lepiej dopuścić kilkuprocentowe odchylenie:

```
having count(*) >=
all (select count(*)*0.95 from ...
```

2. gdy dwie częstości są blisko siebie to dobrym rozwiązaniem jest dopuszczenie odchylenia k wartości

Wartość średnia

- `AVG ([DISTINCT] expr)`

Returns the average value of *expr*. The `DISTINCT` option can be used to return the average of the distinct values of *expr*.

`AVG()` returns `NULL` if there were no matching rows.

```
mysql> SELECT student_name, AVG(test_score)
->      FROM student
->      GROUP BY student_name;
```

[MySQL 5.6 Reference Manual, p. 1372]

C:\Windows\system32\cmd.exe

```
mysql> select avg(indepYear) from Country;
```

avg(indepYear)
1847.2604

1 row in set (0.01 sec)

```
mysql> select sum(indepYear)/count(*) from Country;
```

sum(indepYear)/count(*)
1483.9916

1 row in set (0.00 sec)

```
mysql>
```

Mediana

Jest to wartość, dla której jest dokładnie tyle samo elementów o wartościach od niej niższych, co elementów o wartościach od niej wyższych. Jeśli taka wartość istnieje w zbiorze danych to nazywamy ją medianą statystyczną, jeżeli nie, to oblicza się medianę finansową. Zbiór danych dzieli się wtedy na dwa równoliczne, tak by elementy pierwszego były mniejsze od elementów drugiego, następnie oblicza się średnią między maksimum pierwszego i minimum drugiego.

Inne nazwy: wartość środkowa, wartość przeciętna.

Z medianą statystyczną jest problem, gdy wartości środkowych jest kilka (wskazany wyżej podział jest wówczas niemożliwy).

```
C:\Windows\system32\cmd.exe

mysql> select IndepYear from Country
-> where IndepYear is not NULL
-> order by 1;
+-----+
| IndepYear |
+-----+
| -1523     |
| -1000     |
| -660      |
| 800       |
| 836       |
| 843       |
| 885       |
| 1066      |
| 1991      |
| 1991      |
| 1991      |
| 1991      |
| 1991      |
| 1992      |
| 1993      |
| 1993      |
| 1993      |
| 1994      |
+-----+
192 rows in set (0.01 sec)

mysql>
```



```
C:\Windows\system32\cmd.exe

mysql> select IndepYear from Country
      -> where IndepYear is not NULL
      -> order by 1 limit 96;
+-----+
| IndepYear |
+-----+
|      -1523 |
|      -1000 |
|       -660 |
|         800 |
|         836 |
|          1957 |
|          1958 |
|          1960 |
|          1960 |
|          1960 |
+-----+
96 rows in set (0.00 sec)

mysql>
```

C:\Windows\system32\cmd.exe

```
mysql> select count(*) from Country
-> where IndepYear is not NULL and
-> IndepYear>1960;
```

```
+-----+
| count(*) |
+-----+
|        81 |
+-----+
```

1 row in set (0.00 sec)

```
mysql> select count(*) from Country
-> where IndepYear is not NULL and
-> IndepYear<1960;
```

```
+-----+
| count(*) |
+-----+
|        93 |
+-----+
```

1 row in set (0.00 sec)

```
mysql> select count(*) from Country
-> where IndepYear is not NULL and
-> IndepYear=1960;
```

```
+-----+
| count(*) |
+-----+
|        18 |
+-----+
```

1 row in set (0.00 sec)

```
mysql>
```

Kwartyle

Kwartyle dzielą wszystkie obserwacje na cztery równe co do ilości grupy. Kwartyl pierwszy (Q1) dzieli obserwacje w stosunku 25% do 75%, co oznacza, że 25% obserwacji jest niższa bądź równa wartości Q1, a 75% obserwacji jest równa bądź większa niż Q1.

Kwartyl drugi (Q2), inaczej zwany medianą(!), dzieli obserwacje na dwie części w stosunku 50% do 50%.

Kwartyl trzeci (Q3) dzieli obserwacje w stosunku 75% do 25%, co oznacza, że 75% obserwacji jest niższa bądź równa wartości Q3, a 25% obserwacji jest równa bądź większa niż Q3.

Odchylenie ćwiartkowe (Q), to połowa różnicy między trzecim a pierwszym kwartylem.

Kwantyle

Kwantylem rzędu p z przedziału $(0,1)$ jest taka wartość x_p zmiennej losowej, że wartości $\leq x_p$ są przyjmowane z prawdopodobieństwem co najmniej p , a wartości $\geq x_p$ są przyjmowane z prawdopodobieństwem co najmniej $1-p$.

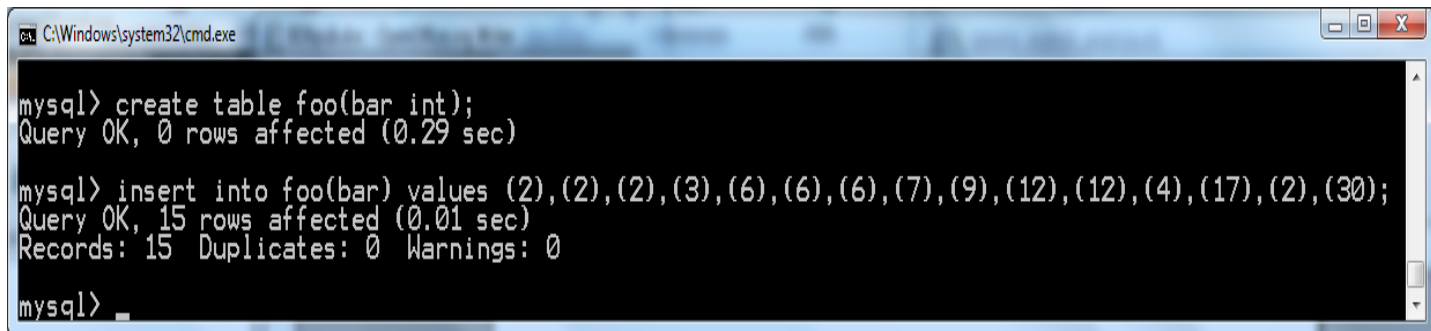
Kwantyl rzędu $\frac{1}{2}$ to inaczej mediana.

Kwantyle rzędu $\frac{1}{4}$, $\frac{2}{4}$, $\frac{3}{4}$ to kwartyle

Kwantyle rzędu $\frac{1}{5}$, $\frac{2}{5}$, $\frac{3}{5}$, $\frac{4}{5}$ to kwintyle.

Kwantyle rzędu $\frac{1}{10}$, $\frac{2}{10}$, ... to decyle.

Przykład



```
C:\Windows\system32\cmd.exe

mysql> create table foo(bar int);
Query OK, 0 rows affected (0.29 sec)

mysql> insert into foo(bar) values (2),(2),(2),(3),(6),(6),(6),(7),(9),(12),(12),(4),(17),(2),(30);
Query OK, 15 rows affected (0.01 sec)
Records: 15  Duplicates: 0  Warnings: 0

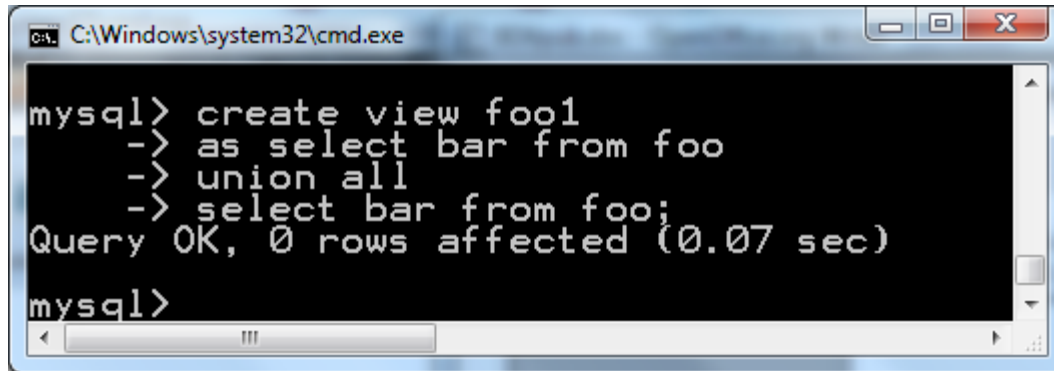
mysql> _
```

```
C:\Windows\system32\cmd.exe

mysql> select * from foo;
+-----+
| bar   |
+-----+
| 2     |
| 2     |
| 2     |
| 3     |
| 6     |
| 6     |
| 6     |
| 7     |
| 9     |
| 12    |
| 12    |
| 4     |
| 17    |
| 2     |
| 30    |
+-----+
15 rows in set (0.00 sec)

mysql>
```

Pierwsza mediana Date'a



```
C:\Windows\system32\cmd.exe

mysql> create view foo1
-> as select bar from foo
-> union all
-> select bar from foo;
Query OK, 0 rows affected (0.07 sec)

mysql>
```

2 2 2 3 6 6 6 7 9 12 12 4 17 2 30

2 2 2 2 3 4 6 6 6 7 9 12 12 17 30

C:\Windows\system32\cmd.exe

```
mysql> create view foo2  
-> as select bar from foo1  
-> where  
-> (select count(*) from foo)<=  
-> (select count(*) from foo1 as f1 where f1.bar>=foo1.bar)  
-> and  
-> (select count(*) from foo)<=  
-> (select count(*) from foo1 as f2 where f2.bar<=foo1.bar);  
Query OK, 0 rows affected (0.10 sec)
```

```
mysql> _
```


C:\Windows\system32\cmd.exe

```
mysql> select * from foo2;
```

bar
6
6
6
6
6
6

6 rows in set (0.02 sec)

```
mysql> select avg(distinct bar) as median  
-> from foo2;
```

median
6.0000

1 row in set (0.02 sec)

```
mysql>
```

C:\Windows\system32\cmd.exe

```
mysql> insert into foo(bar) values (30),(103),(104),(105);
```

Query OK, 4 rows affected (0.01 sec)

Records: 4 Duplicates: 0 Warnings: 0

```
mysql> drop view foo1;
```

Query OK, 0 rows affected (0.03 sec)

```
mysql> drop view foo2;
```

Query OK, 0 rows affected (0.04 sec)

```
mysql> create view foo1
```

-> as select bar from foo

-> union all

-> select bar from foo;

Query OK, 0 rows affected (0.08 sec)

```
mysql> create view foo2
```

-> as select bar from foo1

-> where

-> (select count(*) from foo)<=

-> (select count(*) from foo1 as f1 where f1.bar>=foo1.bar)

-> and

-> (select count(*) from foo)<=

-> (select count(*) from foo1 as f2 where f2.bar<=foo1.bar);

Query OK, 0 rows affected (0.41 sec)

```
mysql> select avg(distinct bar) as median
```

-> from foo2;

+-----+

| median |

+-----+

| 7.0000 |

+-----+

1 row in set (0.02 sec)

```
mysql>
```

2 2 2 2 3 4 6 6 6 7 9 12 12 17 30 30 130 104 105

a teraz dodam 4 i powinno być

2 2 2 2 3 4 4 6 6 6 7 9 12 12 17 30 30 130 104 105

6.5

skrypt.txt

```
insert into foo(bar) value (4);
drop view foo1;
drop view foo2;
create view foo1
  as select bar from foo
  union all
  select bar from foo;
create view foo2
  as select bar from foo1
  where
    (select count(*) from foo)<=
    (select count(*) from foo1 as f1 where f1.bar>=foo1.bar)
  and
    (select count(*) from foo)<=
    (select count(*) from foo1 as f2 where f2.bar<=foo1.bar);
select avg(distinct bar) as median
  from foo2;
```

```
C:\Windows\system32\cmd.exe

mysql> \. skrypt.txt
Query OK, 1 row affected (0.02 sec)
Query OK, 0 rows affected (0.04 sec)
Query OK, 0 rows affected (0.04 sec)
Query OK, 0 rows affected (0.09 sec)
Query OK, 0 rows affected (0.10 sec)

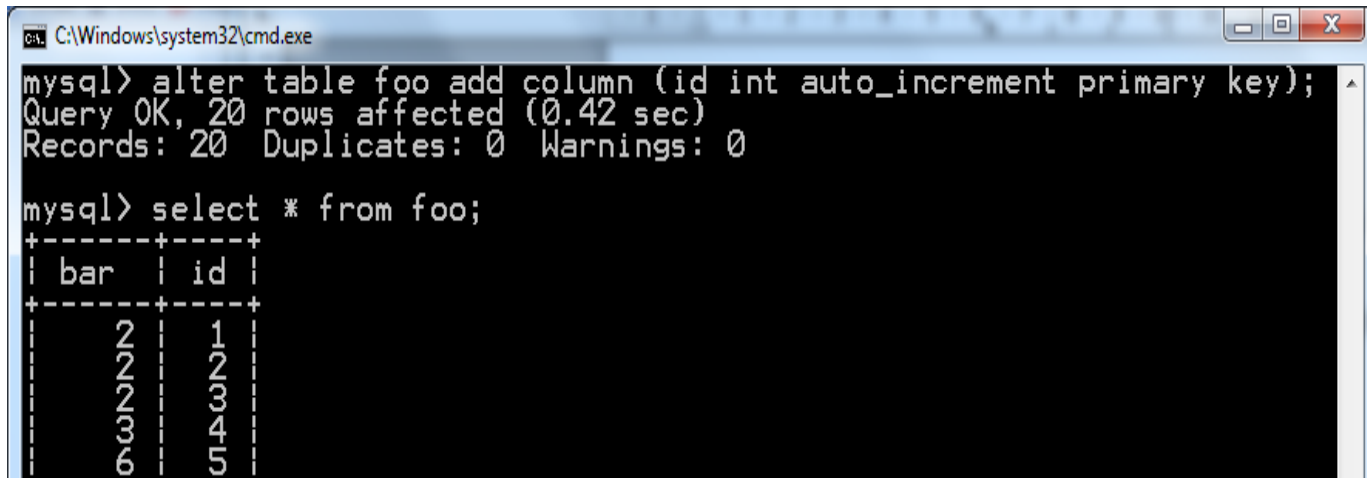
+-----+
| median |
+-----+
| 6.5000 |
+-----+
1 row in set (0.02 sec)

mysql> _
```

Pierwsza mediana Celko

Liczymy osobno najmniejszą wartość w górnej połowie i największą wartość w dolnej połowie.

Operacja grupowania przebiega po wszystkich wierszach tabeli F1 i dlatego należy tabelę FOO uzupełnić o klucz ID:



```
C:\Windows\system32\cmd.exe
mysql> alter table foo add column (id int auto_increment primary key);
Query OK, 20 rows affected (0.42 sec)
Records: 20  Duplicates: 0  Warnings: 0

mysql> select * from foo;
+-----+-----+
| bar   | id   |
+-----+-----+
| 2     | 1    |
| 2     | 2    |
| 2     | 3    |
| 3     | 4    |
| 6     | 5    |
| ...   | ...  |
```

C:\Windows\system32\cmd.exe

```
mysql> select min(bar) from foo where bar in(  
-> select f1.bar from foo as f1, foo as f2  
-> where f2.bar>=f1.bar group by f1.id  
-> having count(*)<=(select ceil(count(*)/2.0) from foo));
```

```
+-----+  
| min(bar) |
```

```
+-----+  
|      7 |
```

```
+-----+  
1 row in set (0.01 sec)
```

```
mysql> select max(bar) from foo where bar in(  
-> select f1.bar from foo as f1, foo as f2  
-> where f2.bar<=f1.bar group by f1.id  
-> having count(*)<=(select ceil(count(*)/2.0) from foo));
```

```
+-----+  
| max(bar) |
```

```
+-----+  
|      6 |
```

```
+-----+  
1 row in set (0.01 sec)
```

```
mysql>
```

Tabela krzyżowa

Tworzymy tabelę pomocniczą a następnie połączymy ją z tabelą danych sprzedaży z czterech lat od 2003 do 2006.

C:\Windows\system32\cmd.exe

```
mysql> create table krzyzowa
```

```
-> (rok int,r1 int,r2 int,r3 int,r4 int,razem int);
```

```
Query OK, 0 rows affected (0.55 sec)
```

```
mysql> insert into krzyzowa values
```

```
-> (2003,1,0,0,0,1),(2004,0,1,0,0,1),(2005,0,0,1,0,1),(2006,0,0,0,1,1);
```

```
Query OK, 4 rows affected (0.01 sec)
```

```
Records: 4 Duplicates: 0 Warnings: 0
```

```
mysql> select * from krzyzowa;
```

rok	r1	r2	r3	r4	razem
2003	1	0	0	0	1
2004	0	1	0	0	1
2005	0	0	1	0	1
2006	0	0	0	1	1

```
4 rows in set (0.00 sec)
```

```
mysql>
```

C:\Windows\system32\cmd.exe

```
mysql> load data infile 'i:/my SQL/mieszk.csv' into table mieszkania  
-> fields terminated by ',' optionally enclosed by '$'  
-> lines terminated by '\r\n';
```

Query OK, 3700 rows affected (0.41 sec)

Records: 3700 Deleted: 0 Skipped: 0 Warnings: 0

```
mysql> select * from mieszkania limit 10;
```

dz	rok	cena	pow
Krzyki	2006	625000	69
Fabryczna	2006	350000	70
Krzyki	2006	338000	41
Fabryczna	2006	380000	72
Srodmiescie	2006	740000	97
Krzyki	2006	386000	62
Krzyki	2006	385000	71
Krzyki	2006	1120919	139
Krzyki	2006	1260381	155
Krzyki	2006	1271240	156

10 rows in set (0.00 sec)

```
mysql>
```

C:\Windows\system32\cmd.exe

```
mysql> select m.dz,  
-> sum(m.pow*k.r1) as "2003",sum(m.pow*k.r2) as "2004",  
-> sum(m.pow*k.r3) as "2005",sum(m.pow*k.r4) as "2006",  
-> sum(m.pow*k.razem) as "2003-2006"  
-> from mieszkania as m,krzyzowa as k  
-> where m.rok=k.rok  
-> group by m.dz order by 6 desc;
```

dz	2003	2004	2005	2006	2003-2006
Krzyki	9737	46396	23839	23646	103618
Srodmiescie	6368	11408	19392	9986	47154
Stare Miasto	8219	12595	5612	9177	35603
Fabryczna	3976	12075	7541	11514	35106
Psie Pole	2334	5304	2576	4296	14510
inne	1591	2196	996	1298	6081

6 rows in set (0.02 sec)

```
mysql>
```

Ćwiczenie:

Wygeneruj wypłaty dla kilku bankomatów dla jednego miesiąca i wykonaj dla takich danych tabelę krzyżową.