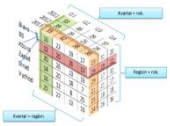
10. Týždeň
Dátové sklady. Dátová kocka.
Pivot tabuľky a pivot kocka.



I Dátové sklady - Data warehouse (DWH) a OLAP

1a DWH a DWH modely

1b Dátové modely pre DWH

Star, Snowflake, DimTable a FactTable

1c Dátová kocka - Data cube

Rozmery, Fakty, Miery

1d OLAP operácie

drill-up a drill-down, slice a dice, pivot

II Pivot tabuľky a pivot kocka.

2a Pivoty v MySQL

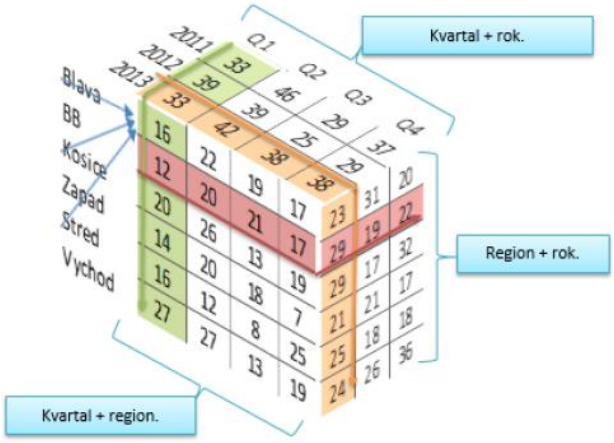
2b Pivoty v R – 1; tidyr::spread, tidyr::pivot_wide

A) Možné prístupy k výpočtu pivot tabuľky pomocou R

B) Pripojenie z R k MySQL a SQL Server

(2c Pivoty v MS Excel

2d Dáta a 3D mapy)



Pivot kocka

I Dátové sklady - Data warehouse (DWH) a OLAP [J. Han, M. Kamber, J. Pei, Data Mining, Concepts and Techniques, 2012, Elsevier]

Dátové sklady vo všeobecnosti poskytujú architektúru a nástroje pre

- systematické usporiadanie dát vo viacrozmernom priestore
- ich analýzu a porozumenie pomocou numericko-štatistických a DB metód

s cieľom dosiahnuť strategické rozhodnutia.

Konštrukcia DWH zahŕňa čistenie, integráciu, transformáciu a modelovanie údajov a predstavuje dôležitý krok predbežného spracovania údajov pred dolovaním dát (data mining DM). Dátové sklady poskytujú

- nástroje na online analytické spracovanie (OLAP) údajov s cieľom interaktívne analyzovať viacrozmerné dáta s rozličnou jemnosťou,

- DM funkcie na klasifikáciu, asociáciu, predpoveď a zoskupovanie.

Kým štatistické DB majú tendenciu zameriavať sa na sociálno-ekonomické aplikácie, OLAP je hlavne zameraný na **podnikové** aplikácie.
Vďaka tomu sa DWH stáva čoraz dôležitejšou platformou pre analýzu údajov, OLAP a DM, a preto DWH predstavuje štandardný krok v procese zisťovania znalostí.

1a DWH a DWH modely

Dátový sklad je sémanticky **konzistentný** sklad údajov a jeho fyzická implementácia z **viacerých** heterogénnych zdrojov slúži ako platforma či dátový model pre rozhodovanie integráciou **rôznych** aplikačných systémov.

Existujú dva štandardné **modely dátového skladu** DWH:
1. **enterprise warehouse** - trvá roky, kým sa vybuduje
2. **data mart** – je to podmnožinou 1. Implementačný cyklus dátového programu sa meria skôr v týždňoch ako v mesiacoch či rokoch. Zdroj údajov môže byť nezávislý od 1, napr. externé / vonkajšie dáta

Ďalej opíšeme dva základné **dátové modely** pre DWH.

1b Dátové modely pre DWH
Star, Snowflake, DimTable a FactTable

Pri návrhu relačných databáz sa štandardne používajú ER diagramy pre dátové modelovanie, v dôsledku čoho schéma databázy pozostáva zo súboru entít a vzťahov medzi nimi. Takýto **relačný** dátový model je vhodný na online **spracovanie transakcií**.
Najobľúbenejší dátový model pre dátový sklad, ktorý uľahčuje online **analýzu údajov** DWH, je **mnohorozmerný** model, ktorý môže existovať vo forme **hviezdnej** schémy (Star), schémy **snehových** vločiek (Snowflake) alebo schémy **galaxie**.

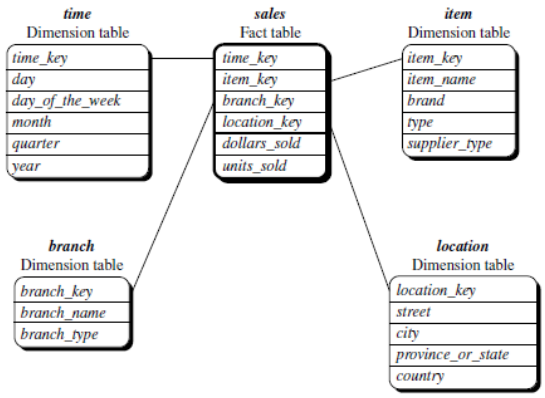


Schéma hviezdy
Najbežnejšou paradigmou modelovania je schéma hviezd, v ktorej sklad údajov obsahuje
- veľkú centrálnu tabuľku (faktická tabuľka) obsahujúcu väčšinu údajov bez redundancií a
- skupinu **menších** sprevádzajúcich tabuliek (tabuľky rozmerov), **jednu pre každý rozmer** (dimension).

Diagram schémy sa podobá hviezde a tabuľky rozmerov sú zobrazené radiálne okolo centrálnej tabuľky faktov.
Zdôrazňujeme, že v schéme hviezdy je každá dimenzia reprezentovaná iba jednou tabuľkou.

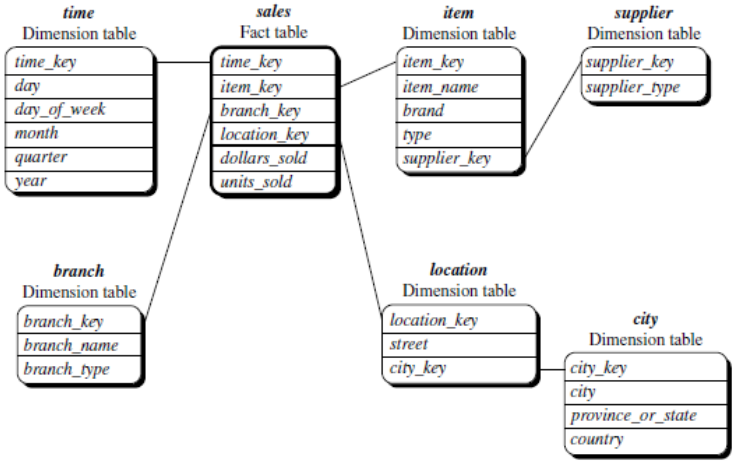


Schéma snehových vločiek
Schéma snehových vločiek je variantom modelu hviezdnej schémy, kde niektoré **tabuľky rozmerov** sú **normalizované**, v dôsledku čoho sa dáta ďalej rozdeľujú do ďalších tabuliek. Výsledný diagram schémy vytvára tvar podobný snehovej vločky.

Schéma galaxie
Sofistikované aplikácie môžu vyžadovať, aby viac tabuliek faktov zdieľali rovnaké tabuľky dimenzií. Na tento druh schémy sa dá pozerať ako na súbor hviezd, a preto sa nazýva schéma galaxie.

1c Dátová kocka - Data cube
Rozmery, Fakty, Miery

Dátová kocka
Dátové sklady a nástroje OLAP sú založené na **viacrozmernom** dátovom modeli. Tento model predstavuje údaje vo forme (často abstraktnej) **dátovej kocky**.
Dátová kocka je metaforou pre viacrozmerné ukladanie údajov. Dátová kocka umožňuje modelovanie a prezeranie údajov vo viacerých dimenziách a je definovaná **rozmermi** (dimensions) a **faktami**.
Rozmery sú *faktory* či ***entity***, v závislosti od ktorých sa vedú záznamy.

Rozmery - Dimensions
Rozlišujeme **hierarchický** H a **nehierarchický** NH rozmer:
Pr.

čas (H), **miesto** (H) vs. **položka** (NH) a **dodávateľ** (NH)

čas: Roky, Kvartaly, Mesiace - hierarchia
miesto: Štáty, Kraje, Okresy, Mestá - hierarchia

Každému rozmeru zodpovedá tzv. tabuľka rozmerov, ktorá podrobnejšie charakterizuje rozmer. Napr. tabuľka rozmerov pre **položku** (TV, mobil, ...) môže obsahovať atribúty názov položky, značku a typ. Tabuľky rozmerov môžu byť zadané buď používatelmi alebo odborníkmi, alebo sa môžu automaticky generovať.

Fakty
Tabuľky rozmerov sú usporiadané okolo tabuľky faktov. Fakty sú **numerické miery**. Tabuľka faktov obsahuje názvy faktov alebo mier, ako aj **kľúče** k jednotlivým tabuľkám rozmerov.

Miera - Measure
Miera dátovej kocky je **numerická funkcia**, ktorú je možné určiť v každom bode v priestore dátovej kocky. Hodnota miery sa vypočíta pre daný bod **agregáciou** údajov zodpovedajúcich príslušným dimenziám, ktoré daný bod definujú.

Mieru je možné zaradiť do troch skupín
- **distribučná**: min, súčet
- **algebraická**: mean, var
- **holistická**: median

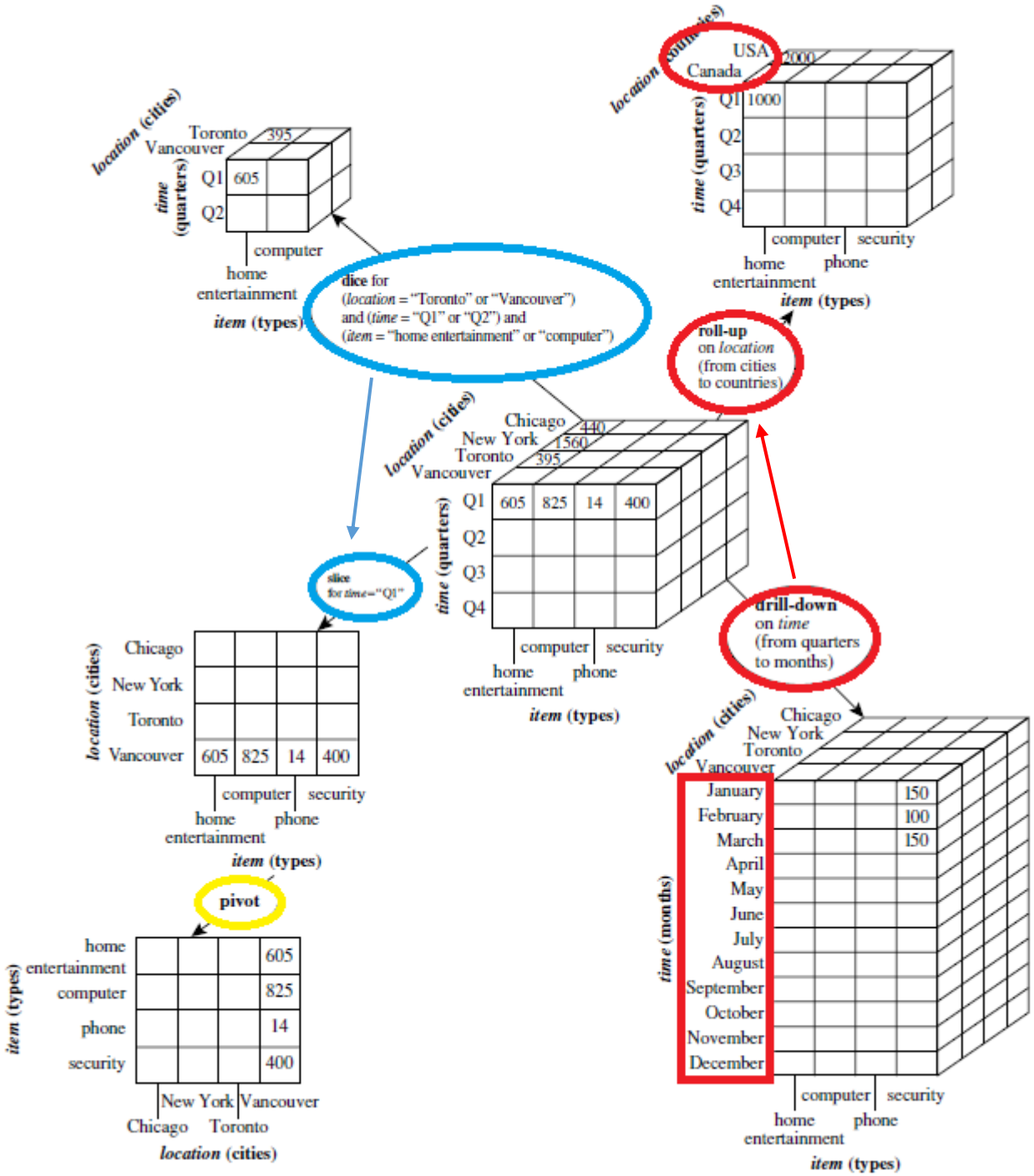
1d OLAP operácie
(roll-up) drill-up a drill-down, slice a dice, pivot

- **drill-up** (nie mesto ale kraj)
- **drill-down** (kvartál a nie rok)
- **slice a dice** Operácia (**slice**) **dice** definuje podmnožinu kocky vykonaním výberu (**iba na jednej** dimenzii) **aspoň na dvoch** dimenziách.

  Príklad Nasl. **d**ice operácia s výberovými kritériami

  {**miesto** Košice alebo Blava} a {**čas** Q1 alebo Q2} a {**položka** počítač alebo práčka}

  narába s tromi rozmermi - miesto, čas, položka.
- **Pivot**



II Pivot tabuľky a pivot kocka

2a Pivoty v MySQL

2b Pivoty v R – 1; tidyr::spread, tidyr::pivot_wide

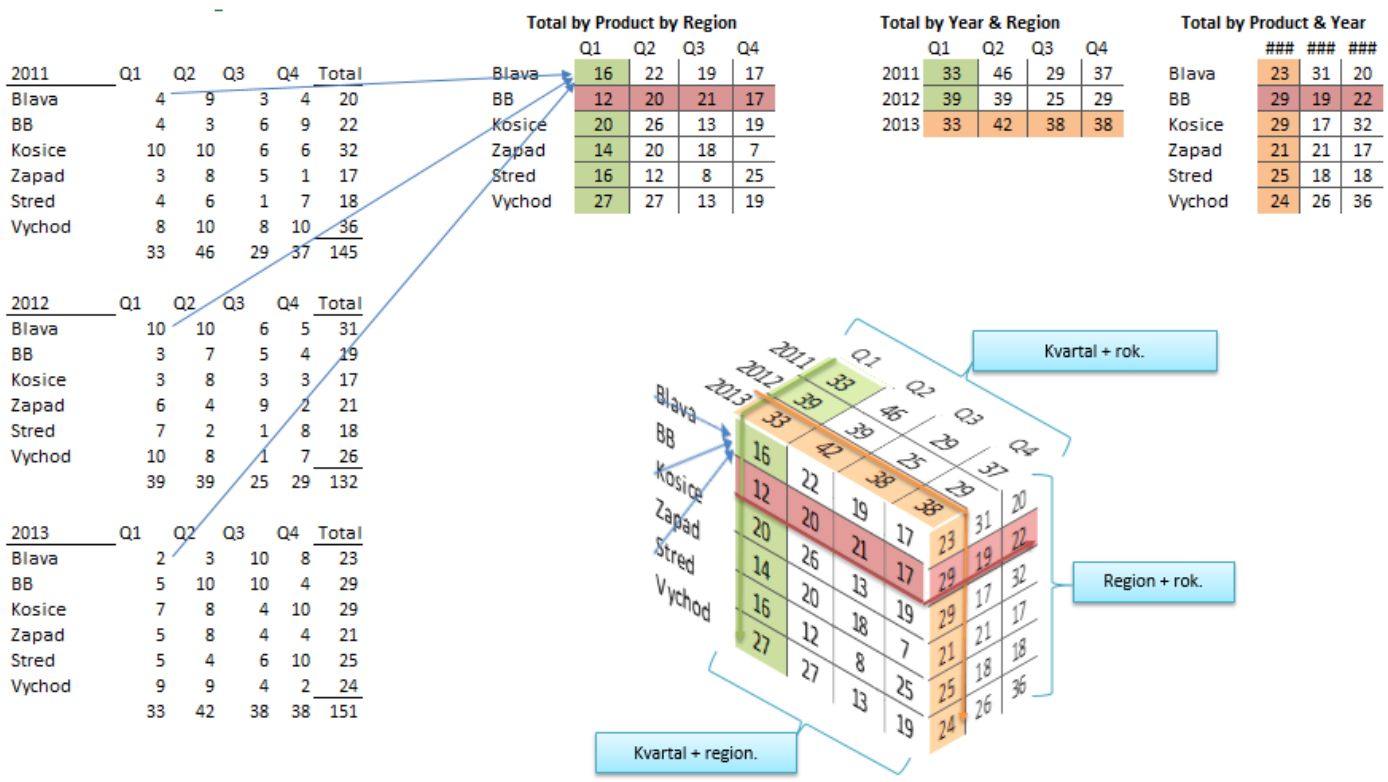
Pripojenie z R k MySQL, SQL Server

(2c Pivoty v MS Excel

2d Dáta a 3D mapy)

Pivot tabuľky a pivot kocka

3 pivot tabuľky



Pivot kocka

2a Pivoty v MySQL

Pivot alebo kontingenčná tabuľka je zovšeobecnením frekvnčnej tabuľky na dvojrozmernú (resp. viacrozmernú) agregačnú tabuľku. V SQL jazykoch ju môžeme vytvoriť pomocou CASE výrazu.

Zistite koľko pacientov navštívilo jednotlivých lekárov v jednotlivých mesiacoch!

Frekvenčná PT -- MySQL, SQL Server:

```
use poliklinika;
SELECT
  month(den) mes
  , COUNT(CASE WHEN idL = 1 THEN 1 ELSE null END) L1
  , COUNT(CASE WHEN idL = 2 THEN 1 ELSE null END) L2
  , COUNT(CASE WHEN idL = 3 THEN 1 ELSE null END) L3
  , COUNT(CASE WHEN idL = 4 THEN 1 ELSE null END) L4
  , COUNT(CASE WHEN idL = 5 THEN 1 ELSE null END) L5
  , SUM(1) AS Vcelku
FROM Navstevy
 GROUP BY month(den);
```

	mes	L1	L2	L3	L4	L5	Vcelku
1	5	0	1	2	0	0	3
2	6	3	0	0	1	0	4
3	7	1	0	2	0	0	3
4	8	0	2	0	1	0	3
5	9	3	0	0	0	1	4
6	10	1	0	0	1	1	3
7	11	1	0	0	0	1	2

```
# <=> (iba MySQL):
```

```
SELECT
  month(den) mes
  , SUM(idL = 1) L1
  , SUM(idL = 2) L2
  , SUM(idL = 3) L3
  , SUM(idL = 4) L4
  , SUM(idL = 5) L5
  , SUM(1) AS Vcelku
FROM Navstevy
 GROUP BY month(den);
```

Zistite sumárne poplatky u jednotlivých lekárov v jednotlivých mesiacoch!
Sumárna PT -- MySQL, SQL Server:

```
SELECT
  month(den) mes
  , SUM(CASE WHEN idL = 1 THEN poplatok ELSE 0 END) L1
  , SUM(CASE WHEN idL = 2 THEN poplatok ELSE 0 END) L2
  , SUM(CASE WHEN idL = 3 THEN poplatok ELSE 0 END) L3
  , SUM(CASE WHEN idL = 4 THEN poplatok ELSE 0 END) L4
  , SUM(CASE WHEN idL = 5 THEN poplatok ELSE 0 END) L5
  , SUM(poplatok) AS Vcelku
FROM Navstevy
 GROUP BY month(den);
```

	mes	L1	L2	L3	L4	L5	Vcelku
1	5	0	0	0	0	0	*NULL*
2	6	900	0	0	500	0	1400
3	7	200	0	1000	0	0	1200
4	8	0	1200	0	800	0	2000
5	9	400	0	0	0	0	400
6	10	300	0	0	800	300	1400
7	11	350	0	0	0	400	750

Namiesto COUNT a SUM je možné použiť ďalšie agregačné funkcie, ako MAX, MIN, AVG, ...

2b Pivoty v R – 1; tidyr::spread, tidyr::pivot_wide

A) Možné prístupy k výpočtu pivot tabuľky pomocou R

Nižšie na úvod do jazyka R ukážeme spolu osem spôsobov (dva z ktorých sú pipe verzie) výpočtu pivot tabuľky priamo v R na jednoduchých dátach z data frame-u *df*. Z nich sa najviac odporúča posledné riešenie pomocou metódy *pivot_wider* balíka *dplyr* a *piping*. Táto metóda dovoľuje skonštruovať aj **komplexnejšie** dopyty na dátových štruktúrach vďaka jej argumentom.

Vytvorme v R jednoduchú dátovú štruktúru typu *data.frame* (⇔ DB tabuľka) s názvom df.

```
df <- read.table(text = "
faktor1 faktor2 x y
AA Red    2 20
AA Blue   3 30
BB Green 1 10
CC Red    2 10
BB Red    3 20
BB Blue   4 21
", header = TRUE);
df;
is.data.frame(df); # TRUE
```

```
  faktor1 faktor2 x  y
1      AA     Red 2 20
2      AA    Blue 3 30
3      BB   Green 1 10
4      CC     Red 2 10
5      BB     Red 3 20
6      BB    Blue 4 21
```

Vytvorme PT početností podľa faktor1 a faktor2.

```
###      1
table(df$faktor1,df$faktor2);
### <=> 2
xtabs(~faktor1 + faktor2, data=df);
### <=> 3
xtabs(~df$faktor1 + df$faktor2);
```

```
> table(df$faktor1,df$faktor2)

     Blue Green Red
  AA    1     0   1
  BB    1     1   1
  CC    0     0   1

> xtabs(~faktor1 + faktor2, data=df)
       faktor2
faktor1 Blue Green Red
     AA    1     0   1
     BB    1     1   1
     CC    0     0   1

> xtabs(~df$faktor1 + df$faktor2)
           df$faktor2
df$faktor1 Blue Green Red
         AA    1     0   1
         BB    1     1   1
         CC    0     0   1
```

Balík tidyverse

```
### <=> 4 - tidyr::spread( dplyr::count ( ...
if (!require(tidyverse, quietly = TRUE))
install.packages("tidyverse");
library(tidyverse);  # => aj dplyr, tidyr
dplyr::count(df,faktor1, faktor2);
# <=> name='n'
dplyr::count(df,faktor1, faktor2, name='n');
# =>
tidyr::spread( dplyr::count (df,faktor1, faktor2),
                  faktor2,  n, fill = 0);

### <=> 5 - spread a pipe – lepsie
df %>%
  count(faktor1, faktor2) %>%
   spread(faktor2, n, fill = 0) ;    #   dcast(faktor1 ~ faktor2,  fill = 0)
```

```
# A tibble: 3 x 4
  faktor1  Blue Green   Red
  <fct>   <dbl> <dbl> <dbl>
1 AA          1     0     1
2 BB          1     1     1
3 CC          0     0     1
```

```
### <=> 6 - summarise(group_by ...
 spread( summarise(group_by( df, faktor1, faktor2), nn = n() ),
            faktor2, nn, fill = 0);

### <=> 7 - summarise(group_by ... - pipe - lepsie
df %>%
 group_by(faktor1, faktor2) %>%
    summarise( nn = n()) %>%
     tidyr::spread (faktor2, nn, fill = 0);
```

Odporúčané riešenie pomocou pivot_wider z tidyr

```
### <=> 8 tidyr::pivot_wider
df %>%
  dplyr::count(faktor1, faktor2) %>%
    tidyr::pivot_wider(
      names_from = faktor2
     ,values_from = n              # n from dplyr::count
     #,values_fill = list(n = 0)    # 0 namiesto NA
     #,values_fn = list(y = sum)  # for uniquely identify
   );
```

```
# A tibble: 3 x 4
  faktor1  Blue   Red Green
  <chr>   <int> <int> <int>
1 AA          1     1     0
2 BB          1     1     1
3 CC          0     1     0
```

Úplne na konci dnešnej prednášky zobrazíme v R pivot tabuľku DF1 na báze v MySQL vytvoreného dopytu ako textového reťazca "Select … CASE …", a na nasledujúcej prednáške vypočítame tabuľku DF1 vlastnými prostriedkami R pomocou metód reshape2::dcast a tidyr::pivot_wider po vysvetlení podstaty poslednej

metódy tidyr::pivot_wider a potom aj metódy tidyr::pivot_longer, z ktorých druhá je inverziou prvej, ale až po detailnejšom úvode do jazyka R a predovšetkým po stručnom vysvetlení systému balíkov **tidyverse**, ktorý zahŕňa aj balík dplyr, a ktorý bol navrhovaný pre manipuláciu s údajmi v rámci data science, data mining či bigdata.

Teraz ukážeme ako sa pripojiť z R k MySQL a SQL Server.

B) Pripojenie z R k MySQL a MS SQL Server

- Balíky: I) RMySQL, II) RODBC

B1) Stručný popis ako sa pripojiť z R k MySQL pomocou balíka RODBC
B2) Stručný popis ako sa pripojiť z R k SQL SERVER pomocou balíka RODBC
B3) Podrobná ukážka dvoch spôsobov ako sa pripojiť z R k MySQL I) a II)
B4) Import textového dopytu z MySQL a SQL SERVER (MS SS) do R

V častiach B1), B2) stručne popíšeme ako sa pripojiť z R k MySQL a MS SQL SERVER pomocou balíka RODBC, potom v časti B3) podrobne ukážeme najprv sprístupnenie databázy *poliklinika* na MySQL z R a potom jej dopytovanie z R na základe balíkov RMySQL a RODBC.

B1) Stručný popis ako sa pripojiť z R k MySQL pomocou balíka RODBC

Ak po inštalácii *odbc mysql driver* a po spustení programu *ODBC Data Sources 64* bit bolo zadané:

Data Source Name: mysql_poliklinika

potom (pozri koniec B3): conn <- odbcConnect('mysql_poliklinika');

B2) Stručný popis ako sa pripojiť z R k SQL SERVER pomocou balíka RODBC

Pripojenie realizujeme pomocou balíka RODBC, ktorý potrebuje štyri údaje pre svoje tri metódy

driver – typ DB server server – názov DB servera DB – názov DB – dopyt	odbcDriverConnect sqlQuery close

Typ a názov servera, a názov DB definujeme raz, ale zadáme dva dopyty qq. Pred volaním dopytu sqlQuery(conn, qq) skôr treba vytvoriť spojenie s DB serverom pomocou metódy odbcDriverConnect a potom ho ukončiť pomocou close.

```
library(RODBC);  # R:
# OK:
cons <- "driver={SQL Server};server=LAPTOP-OH9TOBPI;database=Poliklinika"; cons;
cons <- "driver={SQL Server};server=LAPTOP-OH9TOBPI;database=Poliklinika;trusted_connection=true"; cons;
conn <- odbcDriverConnect(cons);
sqlQuery(conn, 'select idL, krstne, spec from lekari');
close(conn);
```

```
# <=> cons
cons <- "driver={SQL Server};server=LABTOP-OH9TOBPI;database=Poliklinika;trusted_connection=true";
# <=>
dri <- "{SQL Server}";
ser <- "LAPTOP-OH9TOBPI";
DB <- "Poliklinika";
cons1 <- paste0("driver=", dri, ";server=", ser); cons1;
cons2 <- paste0(";database=",DB,";trusted_connection=true"); cons2;
cons <- paste0(cons1, cons2); cons;
```

B3) Podrobná ukážka dvoch spôsobov ako sa pripojiť z R k MySQL I) a II)

Na použitie balíka RMySQL v R na komunikáciu s databázou *poliklinika* z MySQL najprv priamo v R zadefinujeme spojovací objekt s názvom *conMy* pomocou funkcie RMySQL::dbConnect, ktorý názov pri samotnom dopytovaní fetch(dbSendQuery(conMy, ... , ako vidíme, je použitý vo funkcii dbSendQuery.

Na použitie balíka RODBC v R na komunikáciu s databázou *poliklinika* z MySQL najprv mimo R nainštalujeme ODBC mysql driver (tu v autentifikačnom postupe zadáme spojovací názov mysql_poliklinika), ktorý názov pri samotnom dopytovaní sqlQuery(odbcConnect('mysql_poliklinika'), ... je použitý vo funkcii odbcConnect.

I) Pomocou balíka RMySQL bez ODBC

1) (Iba v počítačovej miestnosti) v MySQL Workbenchi spustiť príkaz

```
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY '... heslo1 k DB';
```

2) V Rstudio (HESLO2 je password k MySQL Workbench)

```
if (!require(RMySQL, quietly = TRUE)) install.packages('RMySQL'); # inštaluj RMySQL
library(RMySQL);

conMy <- dbConnect(
  MySQL(),
  user = 'root',
  password = 'Pozor HESLO2 k MySQL Workbench!',
  dbname = 'poliklinika',
  host = '127.0.0.1',
  port = 3306
);
 qL = "select idL, krstne, spec from lekari";
qN = "select idL, poplatok pop, month(den) mes from navstevy";
 Lek <- fetch(dbSendQuery(conMy, qL), n = -1); Lek;
Nav <- fetch(dbSendQuery(conMy, qN), n = -1); Nav;
 on.exit(dbDisconnect(connMy));
```

II) Pomocou balíka RODBC

1) Treba nainstalovat ODBC mysql driver (9.1 alebo 8.0) z
https://dev.mysql.com/downloads/connector/odbc/
Windows (x86, 64-bit), MSI Installer Download + Run Typical

2) Potom spustiť program (Ctrl + Esc): ODBC Data Sources 64 bit
Kliknúť na Add
Zvoliť MySQL ODBC 9.1 Unicode Driver
Vyplniť
Data Source Name: **mysql_poliklinika**
TCP/IP Server: **localhost** Port: 3306
User: **root**
Password: Pozor HESLO2 MySQL !
Database: **poliklinika**
Test => OK OK

3)

```
if (!require(RODBC, quietly = TRUE)) install.packages("RODBC");
library(RODBC);

conn <- odbcConnect('mysqlPoliklinika');
 qL = "select idL, krstne, spec from lekari";
 Lek <- sqlQuery(conn, qL); Lek;
close(conn);
```

B4) Import textového dopytu z MySQL a MS SS do R

Zistite sumárne poplatky u jednotlivých lekárov v jednotlivých mesiacoch!

Tu iba importujeme riešenie z DB do R. Úlohu na báze čistej tabuľky Návstevy pomocou R budeme riešiť na ďalšej prednáške.

```
library(RODBC);     # balik po instalacii sa pripoji
```

```
#qq = "select * from lekari;"
#qq = "select *, month(den) mes from navstevy;";
qq = "SELECT
 month(den) mes
 , SUM(CASE WHEN idL = 1 THEN poplatok ELSE 0 END) L1
 , SUM(CASE WHEN idL = 2 THEN poplatok ELSE 0 END) L2
 , SUM(CASE WHEN idL = 3 THEN poplatok ELSE 0 END) L3
 , SUM(CASE WHEN idL = 4 THEN poplatok ELSE 0 END) L4
 , SUM(CASE WHEN idL = 5 THEN poplatok ELSE 0 END) L5
 , SUM(poplatok) AS Vcelku
 FROM Navstevy
 GROUP BY month(den);
";
```

```
> df # !!!!!
  mes  L1   L2   L3  L4  L5 Vcelku
1   5   0    0    0   0   0     NA
2   6 900    0    0 500   0   1400
3   7 200    0 1000   0   0   1200
4   8   0 1200    0 800   0   2000
5   9 400    0    0   0   0    400
6  10 300    0    0 800 300   1400
7  11 350    0    0   0 400    750
>
```

```
# SQL Server, MySQL
# cons <- "driver={SQL Server};server=LAPTOP-OH9TOBPI;database=Poliklinika"; cons
#conn <- odbcDriverConnect(cons)
conn <- odbcConnect('mysql_poliklinika');
df <- sqlQuery(conn, qq);
close(conn);
df;
```

Zistite počet pacientov u jednotlivých lekárov v jednotlivých mesiacoch!

```
qq = "SELECT
 month(den) mes
 , COUNT(CASE WHEN idL = 5 THEN 1 ELSE null END) Imro
 , COUNT(CASE WHEN idL = 3 THEN 1 ELSE null END) Klara
 , COUNT(CASE WHEN idL = 1 THEN 1 ELSE null END) Oto
 , COUNT(CASE WHEN idL = 2 THEN 1 ELSE null END) Zoli
 , COUNT(CASE WHEN idL = 4 THEN 1 ELSE null END) Zuzka
--  , SUM(1) AS Vcelku
 FROM Navstevy
 GROUP BY month(den);";
# SQL Server, MySQL
#conn <- odbcDriverConnect(cons)
conn <- odbcConnect('mysql_poliklinika');
DF1 <- sqlQuery(conn, qq);
close(conn);
DF1;
# fetch(dbSendQuery(conMy, qq), n = -1)
```

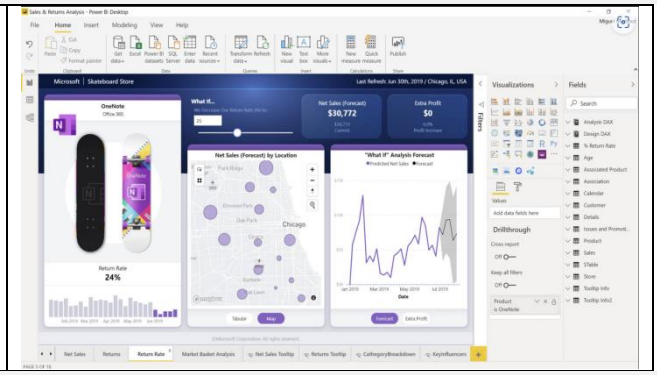
```
> DF1
  mes Imro Klara Oto Zoli Zuzka
1   5    0     2   0    1     0
2   6    0     0   3    0     1
3   7    0     2   1    0     0
4   8    0     0   0    2     1
5   9    1     0   3    0     0
6  10    1     0   1    0     1
7  11    1     0   1    0     0
>
```

(2c Pivoty v MS Excel
2d Dáta a 3D mapy)

	Excel Power BI		Power BI
	Power Query	Power Pivot	Desktop & PowerBI.com
Role	Import and shape data	Data modeling and calculations	Complete business intelligence tool
Language	M	DAX	M and DAX
Key strengths	• Nice easy to use interface • Powerful tools to import and clean data • All Excel users can benefit from this tool	• Easily handle millions of rows of data • Modeling tools for efficient data storage and analysis • Powerful DAX calculations going beyond standard Excel	• Incredible visualization options • Simple built-in interactive options • Powerful DAX calculations • Simple publishing to PowerBI.com and mobile devices

https://www.goskills.com/Excel/Resources/Power-query-vs-power-pivot-power-bi

Excel Power BI vs Power BI Desktop



Power BI Desktop

https://learn.microsoft.com/en-us/power-bi/developer/visualizations/develop-circle-card