7. Týždeň

Systémový pohľad na databázy a tabuľky. Kaskádovité mazanie - integrita dát 3.

A) Databázy – súbory
B) Systémové príkazy o DB a tabuľkách
C) Tabuľky a storage engines
D) Kaskádovité mazanie a aktualizácia - integrita

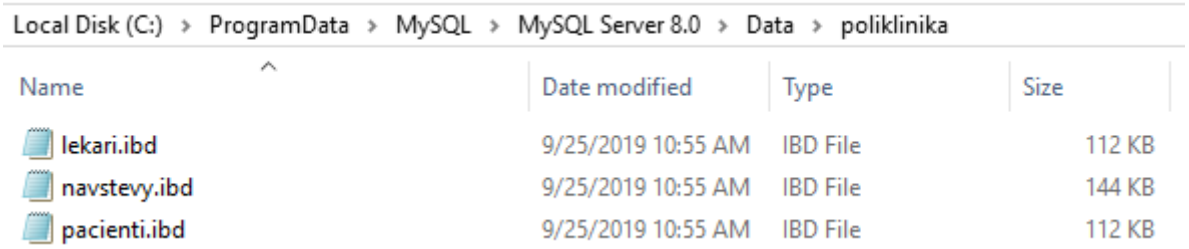
A) Databázy – súbory

Cestu k adresáru, v ktorom sa nachádzajú MySQL databázové údaje zistíme príkazom

```
SHOW VARIABLES WHERE Variable_name = 'datadir';
```

Variable_name	Value
datadir	C:\ProgramData\MySQL\MySQL Server 5.6\Data\

Každej *tabuľke* zodpovedá súbor s koncovkou .ibd.



(Issues with file-based metadata storage included expensive file scans ... The metadata files listed below are removed from MySQL. dbX.opt - collation určuje, ako sú dáta zoradené a porovnávané pomocou priradenia poradia ASCII/ Unicode znakom abecedy.)
Čo obsahuje *ibd* súbor? V MySQL tabuľky štandardne (bez klauzuly ENGINE=) sú *InnoDB* tabuľky, ktoré spolu s indexami sú uložené v systémovom *tablespace* s koncovkou *ibd*. V databázach *tablespace* hrá podobnú úlohu ako priečinok/adresár/folder na pevnom disku počítača.

Teda, .ibd je tabuľkový dátový súbor. *InnoDB* tabuľky a pridružené indexy je možné uložiť vo vlastnom dátovom súbore, *file-per-table tablespaces*. https://dev.mysql.com/doc/refman/5.5/en/innodb-file-per-table-tablespaces.html

B) Systémové príkazy o DB a tabuľkách

`INFORMATION_SCHEMA` tabuľka vs. `SHOW` príkazy

http://dev.mysql.com/doc/refman/5.1/en/information-schema.html http://dev.mysql.com/doc/refman/5.1/en/information-schema-optimization.html http://dev.mysql.com/doc/refman/5.5/en/show.html

- `INFORMATION_SCHEMA` poskytuje prístup k databázovým metaúdajom a obsahuje informácie o DB, tabuľkách, stĺpcoch v MySQL formou *tabuliek*. Obsahuje VIEWs a je iba na čítanie.
- `SHOW` príkazy sú v niektorých prípadoch jednoduchšie, ale menej flexibilné.

Kým `INFORMATION_SCHEMA` je možné `SELECT`-ovať a filtrovať (horizontálne, vertikálne), `SHOW` príkazy je možné kombinovať s klauzulami `LIKE` a `WHERE`.

```
SHOW TABLES FROM INFORMATION_SCHEMA;
SELECT * FROM INFORMATION_SCHEMA.TABLES;
```

Nižšie podrobnejšie popíšeme ako pracovať s `INFORMATION_SCHEMA a SHOW.`

B1) `information_schema`

Informácie o tabuľkách:

```
SELECT schema_name FROM information_schema.schemata;
# SHOW schemas;
```

Informácie o tabuľkách, stĺpcoch a obmedzeniach:

```
SELECT * FROM INFORMATION_SCHEMA.TABLES WHERE
  TABLE_SCHEMA='poliklinika';
SELECT * FROM INFORMATION_SCHEMA.TABLES WHERE
  TABLE_name like 'l%';
SELECT * FROM information_schema.columns
     WHERE TABLE_SCHEMA='poliklinika';
USE INFORMATION_SCHEMA;
SELECT * FROM referential_constraints;
```

B2) `SHOW`

Zoznam existujúcich databáz:

```
SHOW databases; # ⇔ SHOW schemas;
```

Zoznam existujúcich tabuliek v databáze:

```
SHOW tables from poliklinika; -- <=>
USE poliklinika; SHOW tables;

#select * from Poliklinika.Lekari;
SHOW open tables from poliklinika;
SHOW open tables;
```

Zoznam stĺpcov v tabuľke:

```
SHOW COLUMNS FROM lekari;
```

Ďalšie príklady:

```
SHOW tables from information_schema like "%v%";
 =>
SHOW VARIABLES;
SHOW VARIABLES LIKE '%set%';
SHOW VARIABLES WHERE value = 'ON';
SHOW VARIABLES WHERE value = 'OFF';
SHOW VARIABLES WHERE 9 < value and value < 11;

SHOW STATUS LIKE 'Key%';
```

C) Tabuľky a storage engines

Databázový stroj je základná softvérová komponenta DBMS na vytvárenie, čítanie, aktualizovanie a odstránenie dát z databázy.

```
SHOW ENGINES;
```

Engine	Support	Comment	Transactions	XA	Savepoints
FEDERATED	NO	Federated MySQL storage engine	NULL	NULL	NULL
MRG_MYISAM	YES	Collection of identical MyISAM tables	NO	NO	NO
MyISAM	YES	MyISAM storage engine	NO	NO	NO
BLACKHOLE	YES	/dev/null storage engine (anything you write to it disappears)	NO	NO	NO
CSV	YES	CSV storage engine	NO	NO	NO
MEMORY	YES	Hash based, stored in memory, useful for temporary tables	NO	NO	NO
ARCHIVE	YES	Archive storage engine	NO	NO	NO
InnoDB	DEFAULT	Supports transactions, row-level locking, and foreign keys	YES	YES	YES
PERFORMANCE_SCHEMA	YES	Performance Schema	NO	NO	NO

...

```
USE poliklinika;
SHOW CREATE TABLE lekari;
```

=> Open value in Viewer:

```
CREATE TABLE `lekari` (
  `idL` int(11) NOT NULL,
  `krstne` varchar(15) DEFAULT NULL,
  `spec` varchar(20) DEFAULT NULL,
  `datNar` datetime DEFAULT NULL,
  PRIMARY KEY (`idL`)
) ENGINE=InnoDB CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

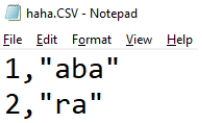
http://dev.mysql.com/doc/refman/5.7/en/innodb-introduction.html http://dev.mysql.com/doc/refman/5.7/en/innodb-storage-engine.html http://dev.mysql.com/doc/refman/5.7/en/storage-engines.html
http://dev.mysql.com/doc/refman/5.7/en/memory-storage-engine.html http://dev.mysql.com/doc/refman/5.7/en/csv-storage-engine.html

MySQL rozlišuje nasledujúce stroje/engines:

- MEMORY Storage Engine
- CSV Storage Engine
  - Comma-separated text file
  - C:\ProgramData\MySQL\MySQL Server 8.0\Data\dbmaz
- **InnoDB** - Transaction s ACID
- MyISAM nepodporuje transakciu. MyISAM umožňuje table lock. Využíva sa pre Web.

```
DROP TABLE IF EXISTS haha;
CREATE TABLE haha (k INT not null, a CHAR(5) not null) ENGINE = CSV;

SHOW CREATE TABLE haha;
INSERT INTO haha VALUES(1,'aba'),(2,'ra');
SELECT * FROM haha;
```



Vytvorená tabuľka štandardne je typu `innoDB`.

```
DROP TABLE IF EXISTS haha2;
CREATE TABLE haha2(x int not null);
insert haha2 values(123);
SELECT * FROM haha2;
SHOW CREATE TABLE haha2; -- innoDB
ALTER TABLE haha2 ENGINE=Memory;
SHOW CREATE TABLE haha2; -- memory

SHOW TABLES;
drop table if exists haha;
drop table if exists haha2;
```

E) Kaskádovité mazanie a aktualizácia - integrita

S cieľom kaskádovitého mazania a aktualizácie je zabezpečiť integritu, konzistenciu dát.

- DB OsobyProjektyVydavky

Osoby

Priezvisko	Krstne
A	a
B	b
C	c

Projekty

Nazov	Veduci
X	B
Y	C
Z	C

Vydaje

Projekt	Polozka	Cena
X	PC	30
X	DVD	2
Y	Cesta	9
Z	Tlaciaren	10
Z	PC	35

Nižšie ilustrujeme niekoľko možností pri navrhovaní tabuliek. Najprv vždy uvedieme príkazy, ktoré môžu mať nežiadúci výsledok (nie nutne chybu) a potom riešenie (resp. návrhové rozhodnutie). Budeme ich označovať *k) resp. *k) .

Nežiadúci výsledok

*1) – vkladáme riadok s hodnotou, ktorá sa odvoláva na neexistujúcu hodnotu
*2) – chceme odstrániť riadok s hodnotou, na ktorú sa odvoláva
*3) – chceme zmeniť hodnotu v riadku, na ktorú sa odvoláva
*4) – nedajú sa odstrániť závislé tabuľky

V ďalšom texte riešenia k daným problémovým miestam sú označené riadkami:

```
#1## *1)A, 2)A Riesenie: FK
#2## *1)A, 2)B Riesenie: FK + kaskadovite mazanie:
#31# *1)A, 2)B, 3)B1 Riesenie: FK + kaskadovite mazanie + kask. aktualizacia mimo CRE.TAB.:
#32# *1)A, 2)B, 3)B1,B2 Riesenie: FK + kaskadovite mazanie + kask. aktualizacia v CRE.TAB.:
### *4) Riesenie - mazat (drop) v spravnom poradi (najprv odstranime tabulku, na ktoru sa
neodvolava, teda Vydaje, ...):

-- *1)  - Vkladáme riadok s hodnotou, ktorá sa odvoláva na neexistujúcu hodnotu.
-- *1)A - Žiaľ prejde a nemal by. Riesenie - FK.

-- *2)  - Chceme odstrániť riadok s hodnotou, na ktorú sa odvoláva.
-- *2)A - Nemal by dovolit. Riesenie - FK.
-- *2)B - Trvam na odstraneni! Riesenie - FK + kaskádovite mazanie.

-- *3)  - Chcem zmenit v riadku hodnotu, na ktorú sa odvoláva.
-- *3)A - Nemal by dovolit. Riesenie - FK.
-- *3)B1- Trvám na zmene. Riesenie - FK + kaskádovita aktualizacia. Mimo CREATE TABLE
-- *3)B2- Trvám na zmene. Riesenie - FK + kaskádovita aktualizacia. Vnutri CREATE TABLE

-- *4)  - Nedajú sa odstrániť závislé tabuľky - kvôli FK.
-- *4)A - Trvám na odstraneni tabulie. Riesenie - zmena poradia droppovania.

### Skopirovat vsetko az do troch tabuliek (obrázok):
DROP   DATABASE IF     EXISTS OsobyProjektyVydavky;
CREATE DATABASE IF NOT EXISTS OsobyProjektyVydavky;

USE OsobyProjektyVydavky;

CREATE TABLE Osoby
(
  idOs       int NOT NULL PRIMARY KEY,
  Priezvisko varchar(20),
  Krstne     varchar(20)
);
```

```
CREATE TABLE Projekty
(
  idPr     int NOT NULL PRIMARY KEY,
  Nazov    varchar(40),
  idOs     int
);

CREATE TABLE Vydaje
(
  idVy     int AUTO_INCREMENT PRIMARY KEY,       -- !!!
  idPr     int,
  Polozka  varchar(30) DEFAULT 'Polozka xyz',    -- !!!
  Cena     double
);
### Riesenie predposledne: 1) 2) a 3)B:
#3B# *1)A, 2)B, 3)B1,B2 Riesenie: FK + kaskadovite mazanie + kask. aktualizacia v CRE.TAB.:
#ALTER TABLE Projekty ADD CONSTRAINT fk_idOs FOREIGN KEY (idOs)
#REFERENCES Osoby(idOs)
#                                  ON DELETE CASCADE
#                                  ON UPDATE CASCADE;
#CREATE TABLE Vydaje(
#  idVy       int AUTO_INCREMENT PRIMARY KEY,            -- !!!
#  idPr       int,
#  Polozka    varchar(30) DEFAULT 'Polozka xyz',   -- !!!
#  Cena       double
# ,CONSTRAINT fk_idPr FOREIGN KEY (idPr) REFERENCES Projekty(idPr)
#                    ON DELETE CASCADE
#                    ON UPDATE CASCADE );

ALTER TABLE Vydaje AUTO_INCREMENT=1;  -- zacni od 1!!!

### Riesenia prve tri: 1) 2) a 3)A:
#1## *1)A, 2)A Riesenie: FK
#ALTER TABLE Projekty ADD CONSTRAINT fk_idOs FOREIGN KEY (idOs) REFERENCES Osoby(idOs);
#ALTER TABLE Vydaje   ADD CONSTRAINT fk_idPr FOREIGN KEY (idPr) REFERENCES Projekty(idPr);

#2## *1)A, 2)B Riesenie: FK + kaskadovite mazanie:
#ALTER TABLE Projekty ADD CONSTRAINT fk_idOs FOREIGN KEY (idOs) REFERENCES Osoby(idOs)
#                                  ON DELETE CASCADE;
#ALTER TABLE Vydaje   ADD CONSTRAINT fk_idPr FOREIGN KEY (idPr) REFERENCES Projekty(idPr)
#                                  ON DELETE CASCADE;

#3A## *1)A, 2)B, 3)B1 Riesenie: FK + kaskadovite mazanie + kask. aktualizacia mimo CRE.TAB.:
#ALTER TABLE Projekty ADD CONSTRAINT fk_idOs FOREIGN KEY (idOs) REFERENCES Osoby(idOs)
#                                  ON DELETE CASCADE
#                                  ON UPDATE CASCADE;
#ALTER TABLE Vydaje   ADD CONSTRAINT fk_idPr FOREIGN KEY (idPr) REFERENCES Projekty(idPr)
#                                  ON DELETE CASCADE
#                                  ON UPDATE CASCADE;

INSERT Osoby VALUES (1, 'A', 'a' );
INSERT Osoby VALUES (2, 'B', 'b' );
INSERT Osoby VALUES (3, 'C', 'c' );

INSERT Projekty VALUES (1, 'X', 2);
INSERT Projekty VALUES (2, 'Y', 3);
INSERT INTO Projekty(idPr, Nazov, idOs)  VALUES (3, 'Z', 3);

INSERT Vydaje(idPr, Polozka, Cena) VALUES (1, 'PC',    30);
INSERT Vydaje(idPr, Polozka, Cena) VALUES (1, 'DVD',    2);
INSERT Vydaje(idPr,          Cena) VALUES (2,           9);
INSERT Vydaje(idPr, Polozka, Cena) VALUES (3, 'Tlaciaren', 10);
INSERT Vydaje(idPr,          Cena) VALUES (3,          35);
```

```
SELECT * FROM Osoby;
SELECT * FROM Projekty;
SELECT * FROM Vydaje;

-- Prejdi k chybam *1)-2), potom 3) a 4) nizsie
```

Osoby

Priezvisko	Krstne
A	a
C	c

Projekty

Nazov	Veduci
Z	C

Vydaje

Projekt	Polozka	Cena
Z	Tlaciaren	10

⇔

	id	Priezvisko	Krstne
1	1	A	a
2	3	C	c

	id	Naz...	idVed
1	3	Z	3

	id	idProj	Polozka	Cena
1	4	3	Tlaciaren	10,00

Po vykonaní nasledovných štyroch (+ ďalších troch SELECT) príkazov výsledkom NIE sú horné 2 x tri dobré (dobré, lebo nie je narušená integrita dát) tabuľky, **ale** dolné tri zlé (zlé, lebo je narušená integrita dát)

```
DELETE FROM Vydaje   WHERE idVy = 5; -- OK.
-- Orig - prve dva problemy: ---------------------------------------
-- *1)  - Vkladáme riadok s hodnotou, ktorá sa odvoláva na neexistujúcu hodnotu.
--         Žiaľ prejde a nemal by.
-- *2)  - Chceme odstrániť riadok s hodnotou, na ktorú sa odvoláva.
--         Nemal by dovolit.
INSERT Projekty VALUES (4, 'X', 5);  -- *1) NO, nehlasi logicku chybu.
DELETE FROM Osoby    WHERE idOs = 2; -- *2) NO, nehlasi logicku chybu.
DELETE FROM Projekty WHERE idPr = 2; -- *2) NO, nehlasi logicku chybu.

SELECT * FROM Osoby;
SELECT * FROM Projekty;
SELECT * FROM Vydaje;
```

idOs	Priezvisko	Krstne
1	A	a
3	C	c
NULL	NULL	NULL

idPr	Nazov	idOs
1	X	2
3	Z	3
4	X	5
NULL	NULL	NULL

idVy	idPr	Polozka	Cena
1	1	PC	30
2	1	DVD	2
3	2	Polozka xyz	9
4	3	Tlaciaren	10
NULL	NULL	NULL	NULL

```
-- 1.riesenie: ----------------------------------------------------
```

Teraz ukážeme prvé riešenie - ZODPOVEDAJÚCE RIADKY VYŠŠIE ZA `AUTO_INCREMENT` a `#1##` ODKOMENTUJTE:

```
#1## *1)A, 2)A Riesenie: FK
ALTER TABLE Projekty ADD CONSTRAINT fk_idOs FOREIGN KEY (idOs) REFERENCES Osoby(idOs);
ALTER TABLE Vydaje   ADD CONSTRAINT fk_idPr FOREIGN KEY (idPr) REFERENCES Projekty(idPr);
```

Po odkomentovaní dvoch riadkov vykonáme všetko vrátane až `DELETE FROM Vydaje WHERE idVy = 5;`

```
-- *1)  - Vkladáme riadok s hodnotou, ktorá sa odvoláva na neexistujúcu hodnotu.
          Neprejde.
-- *2)  - Chceme odstrániť riadok s hodnotou, na ktorú sa odvoláva.
          Neprejde.
```

```
INSERT Projekty VALUES (4, 'X', 5);  -- OK, uz hlasi logicku chybu.
DELETE FROM Osoby    WHERE idOs = 2; -- OK, uz hlasi logicku chybu.
DELETE FROM Projekty WHERE idPr = 2; -- OK, uz hlasi logicku chybu.

SELECT * FROM Osoby;
SELECT * FROM Projekty;
SELECT * FROM Vydaje;
```

Nech trvám na odstránení riadku s hodnotou, na ktorú sa odvoláva, teda na vykonaní dvoch posledných DELETE príkazov.

```
-- 2.riesenie - pokracovanie: ----------------------------------
```

Ukážeme druhé riešenie - ZODPOVEDAJÚCE RIADKY VYŠŠIE ZA `#2##` ODKOMENTUJTE a predchádzajúcom kroku odkomentované dva riadky zakomentujte:

```
#2## *1)A, 2)B Riesenie: FK + kaskadovite mazanie:
ALTER TABLE Projekty  ADD CONSTRAINT fk_idOs FOREIGN KEY (idOs) REFERENCES Osoby(idOs)
                      ON DELETE CASCADE;
ALTER TABLE Vydaje    ADD CONSTRAINT fk_idPr FOREIGN KEY (idPr) REFERENCES Projekty(idPr)
                      ON DELETE CASCADE;
```

Po odkomentovaní štyroch riadkov s `ON DELETE CASCADE` vykonáme znova všetko vrátane až `DELETE FROM Vydaje WHERE idVy = 5;`

```
-- *1)  - Vkladáme riadok s hodnotou, ktorá sa odvoláva na neexistujúcu hodnotu.
          Prejde s kaskadovitym mazanim.
-- *2)  - Chceme odstrániť riadok s hodnotou, na ktorú sa odvoláva.
          Prejde s kaskadovitym mazanim.

INSERT Projekty VALUES (4, 'X', 5);  -- OK, hlasi logicku chybu.
DELETE FROM Osoby    WHERE idOs = 2; -- OK - je zabezpecene kaskadovite mazanie.
DELETE FROM Projekty WHERE idPr = 2; -- OK - je zabezpecene kaskadovite mazanie.
```

Z výsledkov vidíme, že tentokrát prebehli aj kaskádovité mazania:

```
SELECT * FROM Osoby;
SELECT * FROM Projekty;
SELECT * FROM Vydaje;
```

idOs	Priezvisko	Krstne
1	A	a
3	C	c
NULL	NULL	NULL

idPr	Nazov	idOs
3	Z	3
NULL	NULL	NULL

idVy	idPr	Polozka	Cena
4	3	Tlaciaren	10
NULL	NULL	NULL	NULL

```
-- 3.problem: ---------------------------------------------------
-- -- *3)  - Chcem v riadku zmenit hodnotu, na ktorú sa odvoláva.
           Nedovoli.
UPDATE Osoby SET idOs = -1 WHERE idOs = 3;  -- OK, hlasi chybu.
```

Avšak my trváme na zmene. Ako postupovať?

```
-- *3) - Trvám na zmene.
```

Ukážeme tretie riešenie - ZODPOVEDAJÚCE RIADKY VYŠŠIE ZA `#3A##` ODKOMENTUJTE a predchádzajúcom kroku odkomentované štyri riadky zakomentujte:

```
#3A## *1)A, 2)B, 3)B1 Riesenie: FK + kaskadovite mazanie + kask. aktualizacia mimo CRE.TAB.:
ALTER TABLE Projekty ADD CONSTRAINT fk_idOs FOREIGN KEY (idOs) #REFERENCES Osoby(idOs)
                                  ON DELETE CASCADE
                                  ON UPDATE CASCADE;
ALTER TABLE Vydaje    ADD CONSTRAINT fk_idPr FOREIGN KEY (idPr) #REFERENCES Projekty(idPr)
                                  ON DELETE CASCADE
                                  ON UPDATE CASCADE;
```

Alternatívne tretie riešenie ⇔

```
#3B# *1)A, 2)B, 3)B1,B2 Riesenie: FK + kaskadovite mazanie + kask. aktualizacia v CRE.TAB.:
ALTER TABLE Projekty ADD CONSTRAINT fk_idOs FOREIGN KEY (idOs)
REFERENCES Osoby(idOs)
                                  ON DELETE CASCADE
                                  ON UPDATE CASCADE;
```

```
CREATE TABLE Vydaje(
  idVy         int AUTO_INCREMENT PRIMARY KEY,             -- !!!
  idPr         int,
  Polozka      varchar(30) DEFAULT 'Polozka xyz',   -- !!!
  Cena         double
 ,CONSTRAINT fk_idPr FOREIGN KEY (idPr) REFERENCES Projekty(idPr)
                     ON DELETE CASCADE
                     ON UPDATE CASCADE )

-- *3)B1- Trvám na zmene. Riesenie - FK + kaskádovita aktualizacia. Mimo CREATE TABLE
-- *3)B2- Trvám na zmene. Riesenie - FK + kaskádovita aktualizacia. Vnutri CREATE TABLE
```

Z výsledkov vidíme kaskádovitú aktualizáciu (hodnotu -1 v stĺpcoch `idOs` v tabuľkách `Osoby` a `Projekty`) v dvoch tabuľkách:

```
UPDATE Osoby SET idOs = -1 WHERE idOs = 3;  -- OK, kask.update.

SELECT * FROM Osoby;
SELECT * FROM Projekty;
SELECT * FROM Vydaje;
```

idOs	Priezvisko	Krstne
-1	C	c
1	A	a
NULL	NULL	NULL

idPr	Nazov	idOs
3	Z	-1
NULL	NULL	NULL

idVy	idPr	Polozka	Cena
4	3	Tlaciaren	10
NULL	NULL	NULL	NULL

```
-- 4.problem: ------------------------------------------------------
-- *4)  - Nedajú sa odstrániť závislé tabuľky - kvôli FK.
DROP TABLE IF EXISTS Osoby;
DROP TABLE IF EXISTS Projekty;
##DROP TABLE IF EXISTS Vydaje;

### *4) Riesenie - mazat (drop) v spravnom poradi (najprv odstranime tabulku, na ktoru sa
neodvolava, teda Vydaje, ...):
-- *4)A - Trvám na odstraneni tabulie. Riesenie - zmena poradia droppovania.
DROP TABLE IF EXISTS Vydaje;
DROP TABLE IF EXISTS Projekty;
DROP TABLE IF EXISTS Osoby;
```

Záverečný kód (logicky chybný `INSERT` je zakomentovaný):

```
DROP   DATABASE IF     EXISTS OsobyProjektyVydavky;
CREATE DATABASE IF NOT EXISTS OsobyProjektyVydavky;

USE OsobyProjektyVydavky;

CREATE TABLE Osoby
(
  idOs       int NOT NULL PRIMARY KEY,
  Priezvisko varchar(20),
  Krstne     varchar(20)
);

CREATE TABLE Projekty
(
  idPr int NOT NULL PRIMARY KEY,
  Nazov    varchar(40),
  idOs     int
);
```

```
ALTER TABLE Projekty ADD CONSTRAINT fk_idOs FOREIGN KEY (idOs)
REFERENCES Osoby(idOs)
                                ON DELETE CASCADE
                                ON UPDATE CASCADE;
CREATE TABLE Vydaje(
  idVy       int AUTO_INCREMENT PRIMARY KEY,        -- !!!
  idPr       int,
  Polozka    varchar(30) DEFAULT 'Polozka xyz',     -- !!!
  Cena       double,
  CONSTRAINT fk_idPr FOREIGN KEY (idPr) REFERENCES Projekty(idPr)
                ON DELETE CASCADE
                ON UPDATE CASCADE
);

INSERT Osoby VALUES (1, 'A', 'a' );
INSERT Osoby VALUES (2, 'B', 'b' );
INSERT Osoby VALUES (3, 'C', 'c' );

INSERT Projekty VALUES (1, 'X', 2);
INSERT Projekty VALUES (2, 'Y', 3);
INSERT INTO Projekty(idPr, Nazov, idOs)  VALUES (3, 'Z', 3);

INSERT Vydaje(idPr, Polozka, Cena) VALUES (1, 'PC',   30);
INSERT Vydaje(idPr, Polozka, Cena) VALUES (1, 'DVD',   2);
INSERT Vydaje(idPr,          Cena) VALUES (2,         9);
INSERT Vydaje(idPr, Polozka, Cena) VALUES (3, 'Tlaciaren', 10);
INSERT Vydaje(idPr,          Cena) VALUES (3,         35);

DELETE FROM Vydaje WHERE idVy = 5; -- OK.

# INSERT Projekty VALUES (4, 'X', 5); -- OK, hlasi sa chyba.

# Kaskadovite mazanie a aktualizacia:
DELETE FROM Osoby    WHERE idOs = 2;
DELETE FROM Projekty WHERE idPr = 2;
UPDATE Osoby SET idOs = -1WHERE idOs = 3;

SELECT * FROM Osoby;
SELECT * FROM Projekty;
SELECT * FROM Vydaje;
```

idOs	Priezvisko	Krstne
-1	C	c
1	A	a
NULL	NULL	NULL

idPr	Nazov	idOs
3	Z	-1
NULL	NULL	NULL

idVy	idPr	Polozka	Cena
4	3	Tlaciaren	10
NULL	NULL	NULL	NULL

```
### Spravne poradie mazania tabuliek:
DROP TABLE IF EXISTS Vydaje;
DROP TABLE IF EXISTS Projekty;
DROP TABLE IF EXISTS Osoby;
```