1 Tabulky tvorba

1.1 Teorie tvorby tabulek

U tvorby databáze byla zmínka o normalizací tak už víme co bude v tabulkách ;) Než začneme tvořit opět se podíváme co se o hotových tabulkách můžeme dozvědět. Inspirace. Zjistíte že není potřeba vědět co ve sloupci je, ale je potřeba mu říct, zda jde o ***číslo, text, datum***…

1.2 O tabulkách

1.2.1 Přehled hotových tabulek

Syntaxe: Hotové tabulky v databázi

```
SHOW TABLES;
```

1.2.2 Popis tabulky

Syntaxe: struktura tabulky

```
DESCRIBE db;
SHOW COLUMNS FROM db;  -- jiná syntaxe
```

1.2.3 Stav klíčů

Syntaxe: Klíče a indexy … budeme si povídat

```
SHOW KEYS FROM <nazev_tabulky>;
SHOW INDEX FROM <nazev_tabulky>;  -- nebo
```

1.2.4 Odpovídající Engine databáze

Syntaxe: Klíče a indexy … budeme si povídat

```
SELECT table_name FROM INFORMATION_SCHEMA.TABLES
  WHERE engine = 'InnoDB'
```

1.3 Tvorba tabulky

Začneme tabulkou úplně základní a postupně budeme vylepšovat

- Pro tvorbu být své databázi

1.3.1 Teorie – co ej potřeba

Minimálně pro tvorbu tabulky potřebuji

- název tabulky
- název sloupce
- datový typ sloupce

1.3.2 Syntaxe

Syntaxe pro tvorbu, postupně si opět projdeme

```
CREATE [ TEMPORARY ] TABLE [IF NOT EXISTS] table_name
(
  column1 datatype [ NULL | NOT NULL ]
                   [ DEFAULT default_value ]
                   [ AUTO_INCREMENT ]
                   [ UNIQUE KEY | PRIMARY KEY ]
                   [ COMMENT 'string' ],

  column2 datatype [ NULL | NOT NULL ]
                   [ DEFAULT default_value ]
                   [ AUTO_INCREMENT ]
                   [ UNIQUE KEY | PRIMARY KEY ]
                   [ COMMENT 'string' ],
  ...

  | [CONSTRAINT [constraint_name]] PRIMARY KEY [ USING BTREE | HASH ] (index_col_name, ...)

  | [INDEX | KEY] index_name [ USING BTREE | HASH ] (index_col_name, ...)

  | [CONSTRAINT [constraint_name]] UNIQUE [ INDEX | KEY ]
        [ index_name ] [ USING BTREE | HASH ] (index_col_name, ...)

  | {FULLTEXT | SPATIAL} [ INDEX | KEY] index_name (index_col_name, ...)

  | [CONSTRAINT [constraint_name]]
       FOREIGN KEY index_name (index_col_name, ...)
       REFERENCES another_table_name (index_col_name, ...)
       [ MATCH FULL | MATCH PARTIAL | MATCH SIMPLE ]
       [ ON DELETE { RESTRICT | CASCADE | SET NULL | NO ACTION } ]
       [ ON UPDATE { RESTRICT | CASCADE | SET NULL | NO ACTION } ]

  | CHECK (expression)

    {ENGINE | TYPE} = engine_name
  | AUTO_INCREMENT = value
  | AVG_ROW_LENGTH = value
  | [DEFAULT] CHARACTER SET = charset_name
  | CHECKSUM = {0 | 1}
  | [DEFAULT] COLLATE = collation_name
  | COMMENT = 'string'
  | DATA DIRECTORY = 'absolute path'
  | DELAY_KEY_WRITE = { 0 | 1 }
  | INDEX DIRECTORY = 'absolute path'
  | INSERT_METHOD = { NO | FIRST | LAST }
  | MAX_ROWS = value
  | MIN_ROWS = value
  | PACK_KEYS = {0 | 1 | DEFAULT}
  | PASSWORD = 'string'
  | RAID_TYPE = { 1 | STRIPED | RAIDO }
      RAID_CHUNKS = value
      RAID_CHUNKSIZE = value
  | ROW_FORMAT = {DEFAULT | DYNAMIC | FIXED | COMPRESSED}
  | UNION = (table1, ... )
);
```

1.3.3 První tabulka

Do své databáze

Syntaxe: Klíče a indexy … budeme si povídat

```
CREATE TABLE Moje77777 (id INT);
```

Syntaxe: Můžeme se na prázdnou tabulku podívat

```
SELECT * FROM Moje77777;   -- máme jen ID a prázdné
```

1.3.4 Podíváme se na datové typy

Jaký typ zvolit podle údajů v položce (buňce): Co může být v buňce?

- … dotaz …
- Většina položek bude obyčejný text
- číslo domu a psč, výplata budou celá kladná čísla
- Muž žena, kuřák, nekuřák budou mít logickou hodnotu A nebo N.
- …

Dále se dají data omezit a vylepšit (primární klíč atd.) ale o tom později. Podíváme se na datové typy. Přehled máte. Doporučuji se podívat do dokumentace, může být trochu jiný název:

Datový typ	Popis
Int	= celé číslo v rozsahu -2 147 483 648 až 2 147 483 647 … 0 až 4 294 967 295
Smallint	= celé číslo v rozsahu -32 768 až 32 767 …. 0 až 65 535
Tinyint	= celé číslo v rozsahu od -128 do +127 …. 0 až 255
Float	= číslo s pohyblivou řádovou čárkou
decimal(p)	= desetinné číslo s *p* platnými číslicemi
decimal(p, d)	= desetinné číslo s *p* platnými číslicemi a *d* platnými desetinnými místy
Money	= číslo jako peněžní částka (tento typ je snadno nahraditelný např. pomocí *decimal(10, 2)*)
char(x)	= textový řetězec o **pevné** délce *x* znaků (nejvíce však 255)
varchar(x)	= textový řetězec o **variabilní** délce maximálně *x* znaků (nejvíce však 255)
Time	= čas ve formátu HH:MM:SS
Date	= datum ve formátu RRRR-MM-DD
Datetime	= datum a čas ve formátu RRRR-MM-DD HH:MM:SS
Blob	= speciální datový typ pro uložení binárních dat (soubory...)

1.3.4.1 Celá čísla

TINYINT někdy BIT nebo BOOL - rozsah hodnot od -128 do +127, bez znaménka (UNSIGNED) 0 až 255 (zabere 1 byte)
SMALLINT - rozsah hodnot od -32 768 do 32 767, bez znaménka 0 až 65 535 (zabere 2 bytes)
INT nebo INTEGER - rozsah hodnot od -2 147 483 648 do +2 147 483 647, bez znaménka 0 až 4 294 967 295 (zabere 4 bytes)
BIGINT - rozsah hodnot od -9 223 372 036 854 775 808 do +9 223 372 036 854 775 807, bez znaménka (UNSIGNED) tedy 0 až 18 446 744 073 709 551 615 (zabere 8 bytes)

1.3.4.2 Čísla s pohyblivou desetinou čárkou

FLOAT - rozsah hodnot od -3.402823466E+38 do 3.402823466E+38
DOUBLE - rozsah hodnot od -1.7976931348623157E+308 do 1.7976931348623157E+308
DOUBLE PRECISION nebo REAL - synonyma pro typ DOUBLE
DECIMAL(m,d) nebo DEC(m,d) nebo NUMERIC(m,d) - rozsah nastavíme parametry "m" (počet číslic

celkem) a "d" (počet desetinných míst), maximální rozsah je stejný s typem DOUBLE

1.3.4.3 Datum a čas

Na datum a čas se ještě podíváme …

DATE - datum ve formátu "rok-měsíc-den" respektive "RRRR-MM-DD" a v rozsahu 1000-01-01 až 9999-12-31

DATETIME - datum a čas v rozsahu 1000-01-01 00:00:00 až 9999-12-31 23:59:59 (formát je "RRRR-MM-DD HH:MM:SS")

TIMESTAMP(m)- datum a čas v rozsahu 1970-01-01 00:00:00 až 2037-01-01 00:00:00 (vždy se ukládá všech 14 čísel !)

- formát zobrazení (a pro dotazy) provedeme parametrem "m" s hodnotou 14 (nebo chybějící), 12, 10, 8, 6, 4, či 2
- "RRRRMMDDHHMMSS", "RRMMDDHHMMSS", "RRMMDDHHMM", "RRRRMMDD", "RRMMDD", "YYMM", "YY"
- pokud nic nezapíšeme MySQL sám doplní aktuální čas změny v daném řádku

TIME - časový rozsah je od "-838:59:59" do "838:59:59" a formát datového typu "HH:MM:SS"

YEAR(m) - při YEAR(4) rozsah 1901 až 2155, formát je "RRRR", při YEAR(2) bude rozsah 1970-2069

1.3.4.4 Řetězce

CHAR(m) - délka řetězce "m" může být v rozsahu 0-255 – **PEVNÁ VELIKOST**
- pokud je vložený řetězec kratší než nastavíme, chybějící znaky jsou automaticky doplněny mezerami (**má tedy "pevnou" velikost**)
- CHAR (**tedy bez "m"**) je považováno za CHAR(1)

VARCHAR(m) - délka řetězce "m" může být v rozsahu 0-255
- pokud je vložený řetězec kratší než nastavíme, chybějící znaky se nedoplňují (má tedy "plovoucí" velikost), ale navíc se ukládá informace o jeho délce

TINYBLOB nebo TINYTEXT - délka řetězce je maximálně 255 znaků

BLOB nebo TEXT - délka řetězce je maximálně 65 535 znaků

MEDIUMTEXT nebo MEDIUMBLOB - délka řetězce (nebo dat) je maximálně 16 777 215 znaků

LONGTEXT nebo LONGBLOB - délka řetězce (nebo dat) je maximálně 4 294 967 295 znaků

ENUM('prvek1','prvek2',...) - **pole** předem definovaných řetězců (prvků) o maximálním počtu 65 535
- v buňce tabulky pak může být pouze jeden z prvků, které jsem předdefinovali
- místo názvů prvků můžeme používat i jejich pořadí, tedy: 1 (místo 'prvek1'), 2 (místo 'prvek2')...

SET('prvek1','prvek2',...) - **pole** předem definovaných řetězců (prvků) o maximálním počtu 64
- v buňce tabulky pak může být i více z prvků, které jsme předdefinovali

1.3.5 Tabulka – jazyk

Syntaxe:

```
CREATE TABLE `fakulta` (
  `id` int(11) NOT NULL,
)
CHARSET=utf8 COLLATE utf8_czech_ci ;  -- máte v podkladech
```

1.3.6 Tabulka a poznámky

SYNTAXE vytvořit tabulku s poznámkami

```
CREATE TABLE IF NOT EXISTS  `fakulta aa` (
  `id` int(11) NOT NULL COMMENT 'Komentář id',
  `jmeno` varchar(11) COMMENT 'Komentář id'
)
CHARSET=utf8 COLLATE utf8_czech_ci COMMENT = 'Komentář k tabulce';
```

1.3.7 Omezení tabulek - integritní

Dále můžeme chtít, aby nějaká položka měla jedinečnou hodnotu (většinou takovou položkou bude pak primární klíč), nebo měla nějakou hodnotu defaultní, v případě, že ji uživatel nezadal.
Jaká integritní omezení máme k dispozici:

UNIQUE
- v daném sloupci nesmějí být v buňkách stejné hodnoty, tedy co kus to unikát

DEFAULT vychozi_hodnota
- pokud bude buňka prázdná, systém do ní automaticky přiřadí hodnotu "vychozi_hodnota"
- řetězce nezapomeňte psát v uvozovkách

Někdy řazeno jako modifikátory

SYNTAXE

```
CREATE TABLE IF NOT EXISTS Example (
  `id` int(11) UNIQUE,
   `pokus` int(11) DEFAULT 10
);

INSERT INTO Example VALUES (1,2);
INSERT INTO Example (id) VALUES (2);   -- zadá se hodnota 10 do pokus
INSERT INTO Example (id) VALUES (2); xxx zadávám podruhé
```

1.3.8 Další omezení tabulek - modifikátory

AUTO_INCREMENT
- systém si sám ve sloupci generuje unikátní (jedinečné) číselné hodnoty (celočíselný
- (za deklarací nové tabulky můžeme ještě navíc určit výchozí hodnotu: ...AUTO_INCREMENT=50;)

BINARY
- pro CHAR a VARCHAR; tento typ bude brán jako binární a budou se tak rozlišovat malá a velká písmena

FULLTEXT INDEX
- platí pro sloupce typu CHAR, VARCHAR a TEXT
- fultextový index slouží k rychlejšímu hledání dat v textových polích
- hledání v takovýchto polích provádíme pomocí příkazů MATCH a AGAINST
- př.: `SELECT * FROM tabulka WHEREMATCH(sloupec)AGAINST("hledana_hodnota");`

INDEX
- sloupec/sloupce označené jako INDEX umožní rychlejší přístup k datům která obsahují

NOT NULL
- pokud použijeme tento modifikátor, označený typ bude muset v každé buňce obsahovat nějakou hodnotu

NULL - opak NOT NULL; buňka může být prázdná

PRIMARY KEY
- označený typ bude sloužit jako primární klíč - při jeho použití musíme zároveň použít UNIQUE - sloupec nám tedy jedinečným způsobem identifikuje záznamy v tabulce

UNSIGNED
- pokud použijeme modifikátor UNSIGNED, datový typ bude bez znaménka a posune se interval hodnot - u čísel s pohyblivou desetinou čárkou se interval použitím UNSIGNED neposunuje a bereou se jen kladná čísla

ZEROFILL
- použití u čísel, dotaz doplní před číslo nuly v celé jeho šířce IČO
- př.: *pokud máme definováno MEDIUMINT(6) ZEROFILL a je v něm hodnota 123, tak se nám zobrazí 000123*

Syntaxe: Verze 1

```
CREATE TABLE product (
product_id int not null primary key auto_increment,
jmeno varchar(75),
pocet int,
cena decimal(9,2)
);
```

Syntaxe: Verze 2

```
CREATE TABLE `uzivatele` (
  `uzivatele_id` int AUTO_INCREMENT,
  `jmeno` varchar(60),
  `prijmeni` varchar(60),
  `datum_narozeni` date,
  `pocet_clanku` int,
  `typ_clanku` int,
  PRIMARY KEY (`uzivatele_id`)
);
```

1.3.9 Typ tabulky INNODB

Pro pokročilejší správu musím změnit typ tabulky, tj. nemohu využít klasickou MYISAM

```
TYPE=typ_tabulky;
```

- **MYISAM** - standard MySQL od verze 3.23.0; soubory s tabulkami mají koncovku .myd (data) a .myi (indexy)
- ISAM - standardní typ tabulky ve starších databázích; dnes nahrazen typem MYISAM
- MERGE - formát vhodný pro spojení MYISAM tabulek se stejně nadefinovanými poli
- HEAP - tabulka tohoto typu je uložena pouze v paměti (může být velmi rychlá), má ale řadu omezení
- **INNODB** - uzamykání tabulky je vykonáváno na úrovni řádků; před použitím je nutná kompilace MySQL s podporou INNODB
- BDB - typ tabulky podobný INNODB; zatím ve fázi testování

1.4 Cizí klíče

Co se stane, mám fakulty a studenty. Propojeno co když zadám neexistující číslo fakulty? Co když smažu fakultu? Jinak **co bych chtěl aby se stalo?**
Pokud umím nadefinovat tabulku mohu v INNODB využívat ***cizí klíče***.

Než vytvoříme cizí klíče co se bude dít po smazání, co se může dít?

Cizi klíč se muže změnit, pokud se změnil odkaz daného klice, tak se provede:

- ON UPDATE RESTRICT - cizi klic se nemuze změnit na požadavek změny odkazu, tudiz se zmena neprovede
- ON UPDATE CASCADE
- ON DELETE CASCADE - pokud cizi klic ztrati referenci, tak se dany zaznam smaze
- ON DELETE RESTRICT - zaznam s cizim klicem se nemuze smazat na pozadavek odkazu, tudiz se odkaz ani zaznam s cizim klicem nesmaze
- ON DELETE SET NULL - pokud je odkaz smazan, tak se cizi klic nastavi na NULL (pouze, pokud podle definice muze byt sloupec NULL)

1.4.1 Vytvoříme zákazníka ... viz text

SYNTAXE

```
CREATE TABLE zakaznik (
  id INT NOT NULL AUTO_INCREMENT,
  jmeno varchar(50) NOT NULL,
  prijmeni varchar(50) NOT NULL,
  PRIMARY KEY (id)
) ENGINE=INNODB;
```

1.4.2 Vytvoříme kontakt pro zákazníka

SYNTAXE

```
CREATE TABLE kontakt (
  id INT,
  kontakt_id INT,
  adresa varchar(50) NOT NULL,
  type varchar(50) NOT NULL,
  INDEX par_ind (kontakt_id),    -- nazev klíče pro index
  CONSTRAINT CiziKlic_zakaznik FOREIGN KEY (kontakt_id)  --ciziklic musí byt
unikátní název
  REFERENCES zakaznik(id)
  ON DELETE CASCADE
  ON UPDATE CASCADE
) ENGINE=INNODB
```

Syntaxe: Jaké mám indexy

```
SHOW INDEX FROM  kontakt;
```

Nad tabulkou Uzivatele vytvořím INDEX nad sloupcem typ_clanku
Syntaxe:

```
CREATE INDEX IndexClanek ON uzivatele (typ_clanku)
-- bylo vytvořeno v předchozím
```

Kontrola zda se vytvořil

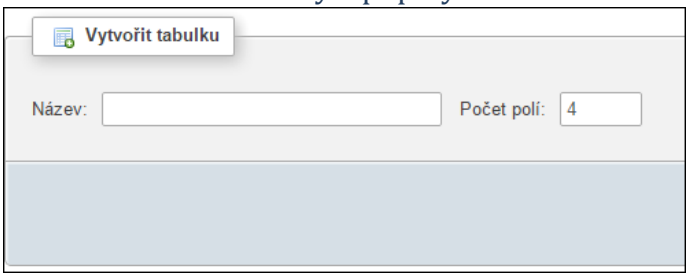
```
SHOW INDEX FROM  uzivatele;
select * from INFORMATION_SCHEMA.KEY_COLUMN_USAGE -- vše
select * from INFORMATION_SCHEMA.KEY_COLUMN_USAGE where TABLE_NAME = 'kontakt';
```

Vypínání klíčů

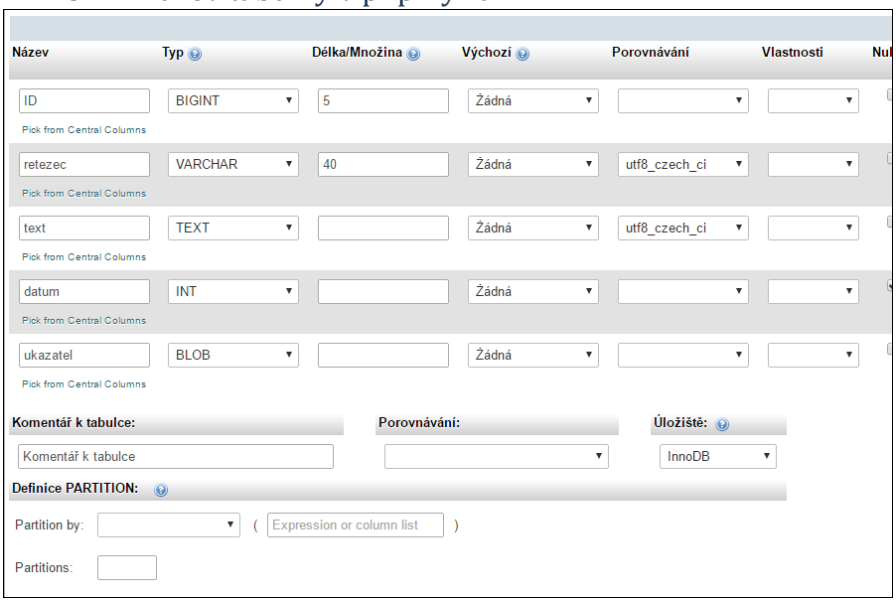
```
SET foreign_key_checks = 0;
....
SET foreign_key_checks = 1;
```

1.5 Tvorba tabulek v phpMyAdmin

1.5.1 Název tabulky v phpMyAdmin



1.5.2 Název tabulky v phpMyAdmin



1.6 Změny v tabulce

Hotové tabulky můžeme měnit

1.6.1 ALTER TABLE syntaxe

```
ALTER TABLE tbl_name
    [alter_specification [, alter_specification] ...]
    [partition_options]

alter_specification:
    table_options
  | ADD [COLUMN] col_name column_definition
        [FIRST | AFTER col_name ]
  | ADD [COLUMN] (col_name column_definition,...)
  | ADD {INDEX|KEY} [index_name]
        [index_type] (index_col_name,...) [index_option] ...
  | ADD [CONSTRAINT [symbol]] PRIMARY KEY
        [index_type] (index_col_name,...) [index_option] ...
  | ADD [CONSTRAINT [symbol]]
        UNIQUE [INDEX|KEY] [index_name]
        [index_type] (index_col_name,...) [index_option] ...
  | ADD FULLTEXT [INDEX|KEY] [index_name]
        (index_col_name,...) [index_option] ...
  | ADD SPATIAL [INDEX|KEY] [index_name]
        (index_col_name,...) [index_option] ...
  | ADD [CONSTRAINT [symbol]]
        FOREIGN KEY [index_name] (index_col_name,...)
        reference_definition
  | ALGORITHM [=] {DEFAULT|INPLACE|COPY}
  | ALTER [COLUMN] col_name {SET DEFAULT literal | DROP DEFAULT}
  | CHANGE [COLUMN] old_col_name new_col_name column_definition
        [FIRST|AFTER col_name]
  | LOCK [=] {DEFAULT|NONE|SHARED|EXCLUSIVE}
  | MODIFY [COLUMN] col_name column_definition
        [FIRST | AFTER col_name]
  | DROP [COLUMN] col_name
  | DROP PRIMARY KEY
  | DROP {INDEX|KEY} index_name
  | DROP FOREIGN KEY fk_symbol
  | DISABLE KEYS
  | ENABLE KEYS
  | RENAME [TO|AS] new_tbl_name
  | RENAME {INDEX|KEY} old_index_name TO new_index_name
  | ORDER BY col_name [, col_name] ...
  | CONVERT TO CHARACTER SET charset_name [COLLATE collation_name]
  | [DEFAULT] CHARACTER SET [=] charset_name [COLLATE [=] collation_name]
  | DISCARD TABLESPACE
  | IMPORT TABLESPACE
  | FORCE
  | {WITHOUT|WITH} VALIDATION
  | ADD PARTITION (partition_definition)
  | DROP PARTITION partition_names
  | DISCARD PARTITION {partition_names | ALL} TABLESPACE
  | IMPORT PARTITION {partition_names | ALL} TABLESPACE
  | TRUNCATE PARTITION {partition_names | ALL}
  | COALESCE PARTITION number
  | REORGANIZE PARTITION partition_names INTO (partition_definitions)
  | EXCHANGE PARTITION partition_name WITH TABLE tbl_name [{WITH|WITHOUT} VALIDATION]
  | ANALYZE PARTITION {partition_names | ALL}
  | CHECK PARTITION {partition_names | ALL}
  | OPTIMIZE PARTITION {partition_names | ALL}
  | REBUILD PARTITION {partition_names | ALL}
  | REPAIR PARTITION {partition_names | ALL}
  | REMOVE PARTITIONING
  | UPGRADE PARTITIONING

index_col_name:
    col_name [(length)] [ASC | DESC]

index_type:
    USING {BTREE | HASH}

index_option:
    KEY_BLOCK_SIZE [=] value
  | index_type
  | WITH PARSER parser_name
  | COMMENT 'string'

table_options:
    table_option [[,] table_option] ...  (see CREATE TABLE options)

partition_options:
    (see CREATE TABLE options)
```

1.6.2 Přejmenování tabulky

Mějme tabulku … máte v podkladech

```
CREATE TABLE `uzivatel_zmeny` (
  `uzivatele_id` int AUTO_INCREMENT,
  `jmeno` varchar(60),
  `prijmeni` varchar(60),
  `datum_narozeni` date,
  `pocet_clanku` int,
  `typ_clanku` int,
  PRIMARY KEY (`uzivatele_id`)
);
```

Přejmenovat tabulku

```
ALTER TABLE `uzivatel_zmeny`   RENAME uzivatel;
```

1.6.3 Nový sloupec

Přidat sloupec nakonec v tabulce …..

```
ALTER TABLE uzivatel ADD COLUMN vydavatel VARCHAR(10);  -- mám přejmenováno
SELECT * FROM uzivat;  -- co se stalo, přidal se vydavatel
```

Přidat za sloupec

```
ALTER TABLE uzivatel ADD COLUMN aaa VARCHAR(10) AFTER jmeno;
SELECT * FROM uzivatel;
XXX  ALTER TABLE uzivatel ADD COLUMN aaa VARCHAR(10) AFTER login; -- není sloupec!!
```

Jako první

```
ALTER TABLE uzivatel ADD COLUMN cislo int FIRST;
SELECT * FROM uzivatel;
```

1.6.4 Smazání sloupce

Smazat sloupec vydavatel

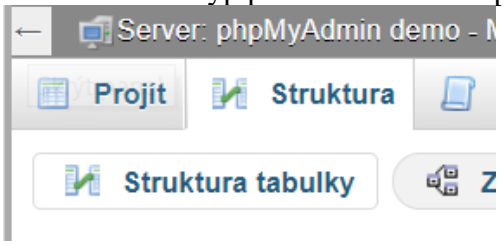
```
ALTER TABLE uzivat DROP vydavatel;
SELECT * FROM uzivat;
```

Smazat sloupec s COLUMN … někdy potřeba

```
ALTER TABLE uzivat DROP COLUMN aaa;
SELECT * FROM uzivat;
```

1.6.5 Změna parametrů a názvů

Podívám se na typ přes Struktura v PhpMyAdmin



Změnit název i typ

```
ALTER TABLE uzivat CHANGE jmeno JmenoCloveka VARCHAR(30) NOT NULL;
DESCRIBE uzivat;
```

Změnit jen typ

```
ALTER TABLE uzivat MODIFY JmenoCloveka VARCHAR(10) NOT NULL;
DESCRIBE uzivat;
```

1.6.6 Smazat INDEXY

Syntaxe: Vytvořit INDEX

```
CREATE INDEX Index_typ ON uzivat (typ_clanku)
DESCRIBE uzivat;
```

Smazat index

```
ALTER TABLE uzivat DROP INDEX Index_typ
DESCRIBE uzivat;
```

1.6.1 Jazyk

Povíme si až budou data v tabulkách

```
ALTER TABLE tbl_name CONVERT TO CHARACTER SET charset_name;
```

1.7 Mazání tabulek

1.7.1 Příkaz TRUNCATE TABLE

- TRUNCATE TABLE je příkaz, který odstraňuje všechny záznamy v tabulce, přičemž definici tabulky samotnou ponechává. Následující dva příkazy budou tedy totožné:
- Z předchozí hodiny si můžete vložit nějakou tabulku s daty ;)

SYNTAXE

```
TRUNCATE TABLE funkce;
```

1.7.2 DROP TABLE

- odstraní z aktivní databáze tabulku s názvem "nazev_tabulky"
- Opravdu jste ve své databázi?

SYNTAXE

```
DROP TABLE osma;
```

1.8 Optimalizace tabulky

Budeme se ještě bavit bude-li na konci čas

SYNTAXE

```
OPTIMIZE TABLE nazev_tabulky;
```

- odstraní z tabulky nepotřebná data po úkonech jako je mazání, rozdělování řádků, opravy v tabulce; dále setřídí indexy a zaktualizuje statistiky.

1.9 Práva nad tabulkami

Opět až probereme uživatele tak si povíme o právech nad tabulkami

1.9.1 Zamykání tabulek

```
LOCK TABLES nazev_tabulky READ, nazev_tabulky WRITE;
```

- uzamkne vyjmenované tabulky pro čtení (READ), nebo zápis (WRITE)
- po uzamknutí mají právo čtení, nebo zápisu v tabulce pouze ty příkazy, které se nachází mezi LOCK ... UNLOCK

```
UNLOCK TABLES;
```

- odemčení všech zamčených tabulek

```
BEGIN; dotaz1; dotaz2; dot...; COMMIT;
```

- pouze u typu tabulky InnoDB - všechny dotazy se vykonají pouze za předpokladu, že se spojení MySQL nepřeruší až do vykonání COMMIT
- pokud je spojení během dotazování přerušeno, neprovede se ani jeden dotaz mezi BEGIN a COMMIT

```
SELECT co_nacist FROM odkud_nacist LOCK IN SHARE MODE;
```

- pouze u typu tabulky InnoDB, - dotaz počká až se dokončí právě probíhající dotazy a až potom načte záznam

1.10 Import / Export

V samostatné hodině, až budou data…

1.11 Dočasné tabulky

Opět v samostatně až budeme umět vkládat i záznamy. V hodině spolu s pohledy, indexy.