MySQL

přes MySQL Command Line Client

Zobrazení existujících databází

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| test               |
+--------------------+
```

Vytvoření databáze

```
mysql> CREATE DATABASE krouzek;
Query OK, 1 row affected (0.00 sec)

mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| krouzek            |
| test               |
+--------------------+

mysql> CREATE DATABASE krouzek;
ERROR 1007 (HY000): Can't create database 'test'; database exists

mysql> CREATE DATABASE IF NOT EXISTS krouzek;
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> CREATE DATABASE nova CHARACTER SET latin2 COLLATE latin2_czech_cs;
Query OK, 1 row affected (0.02 sec)
```

```
set utf8   collate utf8_czech_ci;
set cp1250 collate cp1250_czech_cs;
```

```
CHARSET=UTF-8

=ISO-8859-2
=Windows-1250
```

Odstranění databáze

```
mysql> DROP DATABASE test;
Query OK, 0 rows affected (0.00 sec)

mysql> DROP DATABASE IF EXISTS test;
Query OK, 0 rows affected, 1 warning (0.00 sec)
```

Nastavení aktivní databáze

```
mysql> USE krouzek;
Database changed
```

Nastavení znakové sady pro aktuální práci s databází

```
mysql> CHARSET latin2;

mysql> SET CHARACTER SET latin2;

mysql> SET NAMES latin2;
```

```
latin2/utf8/cp1250/...
```

Zobrazení tabulek v libovolné existující databázi

```
mysql> SHOW TABLES FROM krouzek;
Empty set (0.00 sec)
```

Zobrazení tabulek v používané databázi

```
mysql> SHOW TABLES;
Empty set (0.00 sec)
```

DATOVÉ TYPY

Celá čísla

TINYINT	-128 až +127
TINYINT UNSIGNED	0 až 255
SMALLINT	-32768 až 32767
SMALLINT UNSIGNED	0 až 65535
MEDIUMINT	-8388608 do +8388607
MEDIUMINT UNSIGNED	0 až 16777215
INT nebo INTEGER	-2147483648 až +2147483647
INT UNSIGNED	0 až 4294967295
BIGINT	-9223372036854775808 až +9223372036854775807
BIGINT UNSIGNED	0 až 18446744073709551615
BIT	počet bitů od 0 do 64 Např. BIT (5) – 00000
BOOL nebo BOOLEAN	synonymum pro TINYINT(1) Nula je vyhodnocena jako *false*, ostatní jako *true*

Nepřesná desetinná čísla (zaokrouhlovaná)

FLOAT	-3.402823466E+38 do 3.402823466E+38 Např. FLOAT (10, 5) – až 10 znaků, desetiny zabírají 5 míst
DOUBLE	-1.7976931348623157E+308 do 1.7976931348623157E+308

Přesná desetinná čísla (nezaokrouhlovaná)

DEC nebo DECIMAL (a, b)	ukládá desetinná čísla s pevnou čárkou jako řetězec; rozsah stejný jako u DOUBLE, nastavuje se přes parametry

U typů pracujícími s desetinnými čísly lze použít UNSIGNED. Následně půjdou do sloupce uložit pouze kladná čísla, avšak rozsah kladných hodnot se nezvýší.

Řetězce

CHAR (a)	počet znaků v rozmezí 0 až 255 Pokud je vložen kratší řetězec, chybějící znaky budou mezery CHAR bez určení délky řetězce = CHAR (1)
VARCHAR (a)	počet znaků v rozmezí 0 až 255 U kratšího vloženého řetězce mezery délku nevyplňují
TINYTEXT (a)	délka řetězce je maximálně 255 znaků
TEXT	délka řetězce je maximálně 65535 znaků
MEDIUMTEXT	délka řetězce je maximálně 16777215 znaků
LONGTEXT	délka řetězce je maximálně 4294967295 znaků

BLOB	binární data ukládaná do databáze; rozsah stejný jako u TEXT Stejnou délku řetězce jako TEXT mají i jeho varianty: TINYBLOB, MEDIUMBLOB, LONGBLOB
ENUM ('1', '2'...)	pole předem definovaných řetězců (prvků) o max. počtu 65535 V buňce pak může být pouze jeden z předefinovaných prvků. Místo názvů prvků můžeme používat jejich pořadí.
SET ('1', '2'...)	pole předem definovaných řetězců (prvků) o max. počtu 64 V buňce pak může být více prvků z předem předefinovaných.

Datum a čas

DATE	datum a čas ve formátu RRRR-MM-DD (1000-01-01 až 9999-12-31)
DATETIME	datum a čas ve formátu RRRR-MM-DD HH:MM:SS (1000-01-01 00:00:00 až 9999-12-31 23:59:59)
TIMESTAMP (a)	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 "a" s hodnotou 14 (nebo chybějící), 12, 10, 8, 6, 4, či 2 - "RRRRMMDDHHMMSS", "RRMMDDHHMMSS", "RRMMDDHHMM", "RRRRMMDD", "RRMMDD", "YYMM", "YY" Pokud do buňky tohoto typu nic nezapíšeme MySQL sám doplní aktuální čas změny v daném řádku.
TIME	čas ve formátu HH:MM:SS (rozsah -838:59:59 až 838:59:59)
YEAR (a)	rok ve tvaru RRRR (YEAR(4): 1901 až 2155, YEAR(2): 1970-2069)

MODIFIKÁTORY *(http://mm.gene.cz/#prace_s_daty)*

AUTO_INCREMENT
- systém si sám ve sloupci generuje unikátní (jedinečné) číselné hodnoty
- modifikátor lze použít pouze na celočíselný datový typ
- za deklarací nové tabulky můžeme ještě navíc určit vých. hodnotu: ...AUTO_INCREMENT=5;

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

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

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

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

NOT NULL
- použitím tohoto modifikátoru musí označený typ 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

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

UNSIGNED
- použitím UNSIGNED bude datový typ bez znaménka a posune se interval hodnot
- u čísel s pohyblivou desetinou čárkou se interval použitím UNSIGNED neposunuje a berou se jen kladná čísla

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

Vytvoření tabulky

```
mysql> CREATE TABLE LIDE2 (jmeno VARCHAR(20) NOT NULL, prijmeni VARCHAR(20) NOT
       NULL);

mysql> CREATE TABLE knihy (
    -> nazev VARCHAR(50) NOT NULL,
    -> autor VARCHAR(50) NOT NULL,
    -> rok SMALLINT UNSIGNED DEFAULT 2008 NULL
    -> );

mysql> CREATE TABLE lide (
    -> id INT UNSIGNED NOT NULL AUTO_INCREMENT,
    -> jmeno VARCHAR (20) NOT NULL,
    -> prijmeni VARCHAR (20) NOT NULL,
    -> PRIMARY KEY (id)
    -> );

mysql> CREATE TABLE zamestnanci LIKE lide;
```

Vytvoření dočasné tabulky

Po ukončení spojení tabulka zaniká. Dvě různá připojení mohou vytvořit dvě dočasné tabulky stejného názvu.

```
mysql> CREATE TEMPORARY TABLE osoby LIKE lide;
```

Zobrazení popisu tabulky

```
mysql> DESCRIBE lide;
+----------+------------------+------+-----+---------+----------------+
| Field    | Type             | Null | Key | Default | Extra          |
+----------+------------------+------+-----+---------+----------------+
| id       | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| jmeno    | varchar(20)      | NO   |     |         |                |
| prijmeni | varchar(20)      | NO   |     |         |                |
+----------+------------------+------+-----+---------+----------------+
```

Přejmenování tabulky

```
mysql> ALTER TABLE zamestnanci RENAME zamestnanci_firmy;
```

Podrobné informace o primárních klíčích a indexech

```
mysql> SHOW KEYS FROM lide;

mysql> SHOW INDEX FROM lide;
```

Odstranění tabulky

```
mysql> DROP TABLE zamestnanci_firmy;
Query OK, 0 rows affected (0.00 sec)

mysql> DROP TABLE IF EXISTS zamestnanci_firmy;
Query OK, 0 rows affected, 1 warning (0.00 sec)
```

Přidání sloupce do tabulky

```
mysql> ALTER TABLE lide ADD psc VARCHAR (9);

mysql> ALTER TABLE lide ADD rok SMALLINT UNSIGNED DEFAULT 1990;

mysql> ALTER TABLE lide ADD psc VARCHAR (9);

mysql> ALTER TABLE lide ADD misto VARCHAR (10) AFTER prijmeni;

mysql> ALTER TABLE ADD prvni CHAR (1) AFTER prijmeni;

mysql> ALTER TABLE knihy ADD id_knihy INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY
       KEY FIRST;
```

Odstranění sloupce

```
mysql> ALTER TABLE lide DROP psc;
```

Přejmenování sloupce s úpravou typu

```
mysql> ALTER TABLE lide CHANGE misto mesto VARCHAR (30) NOT NULL;
```

Úprava typu sloupce

```
mysql> ALTER TABLE lide MODIFY mesto VARCHAR (35) NOT NULL;
```

Vkládání záznamů

```
mysql> INSERT INTO lide VALUES (NULL, 'Jiří', 'Novák', 'Brno');

mysql> INSERT INTO lide (jmeno, prijmeni) VALUES ('Karel', 'Veselý');

mysql> INSERT INTO lide (jmeno, prijmeni) VALUES ('Petra', 'Veselá'), ('Ivo',
       'Šťastný');
```

IGNORE

Při pokusu vložit záznamy, kde se kříží zadávaný primární klíč s existujícím vznikne chyba a pokud se vkládá více záznamů, žádný se neuloží. Příklad:

```
mysql> INSERT INTO lide (id, jmeno, prijmeni) VALUES (1, 'Jiří', 'Novák');
ERROR 1062 (23000): Duplicate entry '1' for key 'PRIMARY'
```

```
mysql> INSERT IGNORE INTO lide (id, jmeno, prijmeni) VALUES (1, 'Jiří', 'Novák');
Query OK, 0 rows affected (0.00 sec)
```

S IGNORE se záznam s duplicitním primárním klíčem neuloží, avšak ostatní, které nekolidují, ano.

Aktualizace záznamů

```
mysql> UPDATE lide SET mesto = 'Olomouc';

mysql> UPDATE lide SET rok = rok - 5;
```

Omezení počtu ovlivněných řádků

WHERE

```
mysql> UPDATE lide SET mesto = 'Praha' WHERE jmeno = 'Karel' AND prijmeni =
       'Veselý';

mysql> UPDATE lide SET rok = 1990 WHERE rok < 1990;
```

Používají se logické operátory **AND** (&&), **OR** (||), **NOT** (!),
aritmetické operátory **+** (součet), **-** (odečet), ***** (součin), **/** (podíl), **%** (zbytek po podílu),
a porovnávací operátory **= <>** (!=) **>** **<** **<=** **>=** **<=>** (rovno; včetně hodnot NULL)
a další (viz. později)

LIMIT *od záznamu, počet*

```
mysql> UPDATE lide SET mesto = 'Prostějov' LIMIT 2;
```

U UPDATE pouze *počet*.

Odstranění záznamů

```
mysql> DELETE FROM lide;

mysql> DELETE FROM lide WHERE id = 4;

mysql> TRUNCATE TABLE knihy; (rychlejší než DELETE FROM knihy;)
```

Výpis záznamů

```
mysql> SELECT * FROM lide;
+----+-------+----------+-----------+------+
| id | jmeno | prijmeni | mesto     | rok  |
+----+-------+----------+-----------+------+
|  1 | Jiří  | Novák    | Prostějov | 1990 |
|  2 | Karel | Veselý   | Prostějov | 1991 |
|  3 | Petra | Veselá   | Olomouc   | 1990 |
+----+-------+----------+-----------+------+

mysql> SELECT * FROM telefony WHERE cislo <> 0;
+----------+-----------+
| id_osoby | cislo     |
+----------+-----------+
|        1 | 543111111 |
|        1 | 543222222 |
|        2 | 543222222 |
+----------+-----------+

mysql> SELECT * FROM telefony LIMIT 1;
+----------+-----------+
| id_osoby | cislo     |
+----------+-----------+
|        1 | 543111111 |
+----------+-----------+
```

BETWEEN a NOT BETWEEN (hodnoty v rozmezí)

```
mysql> SELECT * FROM lide WHERE rok BETWEEN 1991 AND 2000;
+----+-------+----------+-----------+------+
| id | jmeno | prijmeni | mesto     | rok  |
+----+-------+----------+-----------+------+
|  2 | Karel | Veselý   | Prostějov | 1991 |
+----+-------+----------+-----------+------+

mysql> SELECT * FROM lide WHERE rok NOT BETWEEN 1991 AND 2000;
+----+-------+----------+-----------+------+
| id | jmeno | prijmeni | mesto     | rok  |
+----+-------+----------+-----------+------+
|  1 | Jiří  | Novák    | Prostejov | 1990 |
|  3 | Petra | Veselá   | Olomouc   | 1990 |
+----+-------+----------+-----------+------+
```

IN a NOT IN (hodnoty z výčtu)

```
mysql> SELECT * FROM lide WHERE rok IN (1988, 1989, 1990);
+----+-------+----------+-----------+------+
| id | jmeno | prijmeni | mesto     | rok  |
+----+-------+----------+-----------+------+
|  1 | Jiří  | Novák    | Prostějov | 1990 |
|  3 | Petra | Veselá   | Olomouc   | 1990 |
+----+-------+----------+-----------+------+
```

IS NULL a IS NOT NULL

```
mysql> SELECT * FROM lide WHERE prijmeni IS NULL;
Empty set (0.00 sec)
```

LIKE a NOT LIKE

```
mysql> SELECT * FROM lide WHERE prijmeni LIKE 'Ves%';
+----+-------+----------+-----------+------+
| id | jmeno | prijmeni | mesto     | rok  |
+----+-------+----------+-----------+------+
|  2 | Karel | Veselý   | Prostějov | 1991 |
|  3 | Petra | Veselá   | Olomouc   | 1990 |
+----+-------+----------+-----------+------+

mysql> SELECT * FROM lide WHERE prijmeni LIKE 'Vesel_';
+----+-------+----------+-----------+------+
| id | jmeno | prijmeni | mesto     | rok  |
+----+-------+----------+-----------+------+
|  2 | Karel | Veselý   | Prostějov | 1991 |
|  3 | Petra | Veselá   | Olomouc   | 1990 |
+----+-------+----------+-----------+------+
```

REGEXP (RLIKE) (regulární výrazy)

```
mysql> SELECT * FROM lide WHERE prijmeni REGEXP 'á$';
+----+-------+----------+---------+------+
| id | jmeno | prijmeni | mesto   | rok  |
+----+-------+----------+---------+------+
|  3 | Petra | Veselá   | Olomouc | 1990 |
+----+-------+----------+---------+------+
```

spojení tabulek bez pomoci JOIN

```
mysql> SELECT * FROM lide, telefony WHERE lide.id = telefony.id_osoby;
+----+-------+----------+-----------+------+----------+-----------+
| id | jmeno | prijmeni | mesto     | rok  | id_osoby | cislo     |
+----+-------+----------+-----------+------+----------+-----------+
|  1 | Jiří  | Novák    | Prostějov | 1990 |        1 | 543111111 |
|  1 | Jiří  | Novák    | Prostějov | 1990 |        1 | 543222222 |
|  2 | Karel | Veselý   | Prostějov | 1991 |        2 | 543222222 |
+----+-------+----------+-----------+------+----------+-----------+
```

*Vybrány byly pouze ty záznamy, které byly podle ID nalezeny v **obou** tabulkách (bez Petry Veselé)*

spojení tabulek pomocí JOIN

```
mysql> SELECT * FROM lide JOIN telefony ON lide.id = telefony.id_osoby;
+----+-------+----------+-----------+------+----------+-----------+
| id | jmeno | prijmeni | mesto     | rok  | id_osoby | cislo     |
+----+-------+----------+-----------+------+----------+-----------+
|  1 | Jiří  | Novák    | Prostějov | 1990 |        1 | 543111111 |
|  1 | Jiří  | Novák    | Prostějov | 1990 |        1 | 543222222 |
|  2 | Karel | Veselý   | Prostějov | 1991 |        2 | 543222222 |
+----+-------+----------+-----------+------+----------+-----------+

mysql> SELECT cislo FROM telefony JOIN lide ON lide.id = telefony.id_osoby
       WHERE id = 1;
+-----------+
| cislo     |
+-----------+
| 543111111 |
| 543222222 |
+-----------+

mysql> SELECT * FROM lide LEFT JOIN telefony ON lide.id = telefony.id_osoby;
+----+-------+----------+-----------+------+----------+-----------+
| id | jmeno | prijmeni | mesto     | rok  | id_osoby | cislo     |
+----+-------+----------+-----------+------+----------+-----------+
|  1 | Jiří  | Novák    | Prostějov | 1990 |        1 | 543111111 |
|  1 | Jiří  | Novák    | Prostějov | 1990 |        1 | 543222222 |
|  2 | Karel | Veselý   | Prostějov | 1991 |        2 | 543222222 |
|  3 | Petra | Veselá   | Olomouc   | 1990 |     NULL |      NULL |
+----+-------+----------+-----------+------+----------+-----------+
```

Vybrány byly pouze všechny záznamy z LIDE, bez ohledu na to, jestli k nim byly v TELEFONY nalezeny záznamy

```
mysql> SELECT * FROM lide RIGHT JOIN telefony ON lide.id = telefony.id_osoby;
+------+-------+----------+-----------+------+----------+-----------+
| id   | jmeno | prijmeni | mesto     | rok  | id_osoby | cislo     |
+------+-------+----------+-----------+------+----------+-----------+
|    1 | Jiří  | Novák    | Prostějov | 1990 |        1 | 543111111 |
|    1 | Jiří  | Novák    | Prostějov | 1990 |        1 | 543222222 |
|    2 | Karel | Veselý   | Prostějov | 1991 |        2 | 543222222 |
+------+-------+----------+-----------+------+----------+-----------+
```

Vybrány byly pouze všechny záznamy z TELEFONY, bez ohledu na to, jestli k nim byly v LIDE nalezeny záznamy

AS (zkrácené pojmenování tabulky)

```
mysql> SELECT t.cislo FROM telefony AS t;
```

DISTINCT (odstranění duplikátů ve výpisu)

```
mysql> SELECT DISTINCT cislo FROM telefony;
+-----------+
| cislo     |
+-----------+
| 543111111 |
| 543222222 |
+-----------+
```

ORDER BY (řazení)

```
mysql> SELECT * FROM lide ORDER BY prijmeni, jmeno;
+----+-------+----------+-----------+------+
| id | jmeno | prijmeni | mesto     | rok  |
+----+-------+----------+-----------+------+
|  1 | Jiří  | Novák    | Prostějov | 1990 |
|  2 | Karel | Veselý   | Prostějov | 1991 |
|  3 | Petra | Veselá   | Olomouc   | 1990 |
+----+-------+----------+-----------+------+

mysql> SELECT * FROM lide ORDER BY prijmeni, jmeno DESC;
+----+-------+----------+-----------+------+
| id | jmeno | prijmeni | mesto     | rok  |
+----+-------+----------+-----------+------+
|  3 | Petra | Veselá   | Olomouc   | 1990 |
|  2 | Karel | Veselý   | Prostějov | 1991 |
|  1 | Jiří  | Novák    | Prostějov | 1990 |
+----+-------+----------+-----------+------+

mysql> SELECT * FROM lide ORDER BY RAND() LIMIT 1;

Vybere např.

+----+-------+----------+-----------+------+
| id | jmeno | prijmeni | mesto     | rok  |
+----+-------+----------+-----------+------+
|  1 | Jiří  | Novák    | Prostějov | 1990 |
+----+-------+----------+-----------+------+
```

CASE ... END

```
mysql> SELECT prijmeni, CASE rok WHEN 1990 THEN 'narozen roku 1990' ELSE
       'nenarozen roku 1990' END FROM lide;
+----------+-----------------------------------------------------------------------------+
| prijmeni | CASE rok WHEN 1990 THEN 'narozen roku 1990' ELSE 'nenarozen roku 1990' END |
+----------+-----------------------------------------------------------------------------+
| Novák    | narozen roku 1990                                                           |
| Veselý   | nenarozen roku 1990                                                         |
| Veselá   | narozen roku 1990                                                           |
+----------+-----------------------------------------------------------------------------+
```

Manipulace s čísly

```
mysql> SELECT MAX(rok) FROM lide;
+----------+
| MAX(rok) |
+----------+
|     1991 |
+----------+
```

AVG(sloupec)	Spočítá průměr číselných hodnot ve sloupci
COUNT(sloupec)	Spočítá počet hodnot ve sloupci
COUNT(**DISTINCT** sloupec)	Spočítá počet jedinečných hodnot ve sloupci
GREATEST(sloupec1, sl2, sl3)	Vrátí největší hodnotu ze zadaných sloupců
LEAST(sloupec1, sl2, sl3)	Vrátí nejmenší hodnotu ze zadaných sloupců
MAX(sloupec)	Vybere záznamy s největší hodnotou
MIN(sloupec)	Opak MAX()
MOD(sloupec dělenec, sloupec dělitel)	Vrátí zbytek po dělení
ROUND(číslo, počet míst)	Zaokrouhlí číslo na celé číslo (druhý parametr není povinný)
STD(sloupec)	Spočítá směrodatnou odchylku číselných hodnot ve sloupci
SUM(sloupec)	Sečte všechny číselné hodnoty ve sloupci

Manipulace s textem

```
mysql> SELECT LOWER(jmeno), UPPER(prijmeni) FROM lide;
+--------------+-----------------+
| LOWER(jmeno) | UPPER(prijmeni) |
+--------------+-----------------+
| jiří         | NOVÁK           |
| karel        | VESELÁ          |
| petra        | VESELÁ          |
+--------------+-----------------+
```

LENGTH(sloupec)	Vrátí délku řetězce ve sloupci
LOCATE(retezec , sloupec, start)	Hledá ve "sloupec" hodnotu "retezec", začíná na pozici "start"
SUBSTRING(sloupec, start)	Vypíše řetězec ze "sloupec", bude začínat od pozice "start"
REPLACE(sloupec, nahradit, cim)	V řetězci ve "sloupec" hledá "nahradit", a ve výpisu jej nahradí za řetězec "cim"
REVERSE(sloupec)	Řetězec vrátí otočený
TRIM(sloupec)	Odstraní z řetězce mezery na jeho začátku i konci
TRIM(**LEADING** 'a' FROM sloupec)	Odstraní řetězec ('a') ze začátku řetězce ve 'sloupec'
TRIM(**TRAILING** 'a' FROM sloupec)	Odstraní řetězec ('a') z konce řetězce ve 'sloupec'
TRIM(**BOTH** 'a' FROM sloupec)	Odstraní řetězec ('a') ze začátku i z konce řetězce ve 'sloupec'
LTRIM(sloupec)	Odstraní z řetězce mezery na jeho začátku
RTRIM(sloupec)	Odstraní z řetězce mezery na jeho konci
UPPER(sloupec)	Převede všechna písmena řetězce na velká
LOWER(sloupec)	Převede všechna písmena řetězce na malá

Manipulace s časem

```
mysql> SELECT NOW();
+---------------------+
| now()               |
+---------------------+
| 2008-01-26 17:59:46 |
+---------------------+
```

SELECT NOW(); — Vrátí aktuální datum a čas ve formátu RRRR-MM-DD HH:MM:SS

SELECT CURRENT_DATE(); — Vrátí aktuální datum ve formátu RRRR-MM-DD

SELECT CURRENT_TIME(); — Vrátí aktuální čas ve formátu HH:MM:SS

SELECT DATE_FORMAT(*vstup*, *výstup*); — Vrátí čas v určeném tvaru

```
mysql> SELECT DATE_FORMAT(NOW(),"%Y");
+-------------------------+
| DATE_FORMAT(NOW(),"%Y") |
+-------------------------+
| 2008                    |
+-------------------------+
```

%Y	rok (RRRR)	(např. 2008)
%y	rok (RR)	(např. 08)
%m	měsíc (MM)	(např. 08, 09, 10)
%c	měsíc (M nebo MM)	(např. 8, 9, 10)
%M	název měsíce	(např. January)
%b	zkr. název měsíce	(např. Jan)
%u	číslo týdne v roce	(např. 04)
%D	den řadovou číslovkou	(např. 26th)
%d	den v měsíci (DD)	(např. 01, 02)
%e	den v měsíci (D či DD)	(např. 1, 2)
%w	číslo dne v týdnu (D)	(např. 0, 6)
%W	název dne v týdnu	(např. Monday)
%a	zkr. název dne v týdnu	(např. Mon)
%j	číslo dne v roce (DDD)	(např. 026)
%H	hodina (HH)	(např. 00, 15)
%k	hodina (H nebo HH)	(např. 0, 1)
%h	hodina ve 12hodinovém formátu (HH)	
%l	hodina ve 12hodinovém formátu (H)	
%i	minuty (MM)	(např. 09, 10)
%s	sekundy (SS)	(např. 09, 10)

SELECT QUARTER(NOW()); — Vrátí číslovku určující aktuální čtvrtletí

Výpis záznamů do souboru

```
mysql> SELECT * INTO OUTFILE 'vystup.txt' FIELDS TERMINATED BY ';' FROM lide;
```

Soubor je standartně uložen do stejného umístění, kam se ukládají soubory databáze