2. Týždeň

MySQL - dátové typy, operátory; Funkcie numerické, reťazcové a časové

1. **Prvky jazyka MySQL** http://dev.mysql.com/doc/refman/5.7/en/language-structure.html
2. **Typy a pretypovanie** http://dev.mysql.com/doc/refman/5.7/en/data-types.html
3. **Operátory** http://dev.mysql.com/doc/refman/5.0/en/non-typed-operators.html
4. **Numerické funkcie** http://dev.mysql.com/doc/refman/5.1/en/numeric-functions.html
5. **Reťazcové funkcie** https://dev.mysql.com/doc/refman/5.1/en/string-functions.html
6. **Dátumy a časové funkcie** https://dev.mysql.com/doc/refman/5.1/en/date-and-time-functions.html

1. Prvky jazyka MySQL http://dev.mysql.com/doc/refman/5.7/en/glossary.html

Jazyk MySQL sa skladá z dvoch celkov. Prvý celok tvoria príkazy DDL, DML a SELECT.
Druhý celok MySQL tvoria bežné prvky programovacích jazykov:

- identifikátory – stĺpcov, názvov Serverov, DB, Tab, pohľadov
- premenné – `@x`
- typy dát (dátových objektov) - stĺpcov, premenných, parametrov
- výrazy - skalárne: konštanty, sk. funkcie, odkaz na stĺpec, premenné
- operátory - pre výrazy; logické, porovnávacie, ...
- kľúčové slová
- komentáre
- funkcie - SUM, SUBSTRING, ...

Nižšie sa podrobnejšie venujeme dátovým typom a niektorým štandardným funkciám MySQL, ktoré ilustrujeme aj príkladmi.

2. Typy a pretypovanie

Typy

Dátové typy aj v jazyku SQL zohrávajú dôležitú úlohu. Nastavenie nesprávneho typu stĺpca môže mať za následok pomalšiu odozvu dopytov, ale predovšetkým neefektívne využitie hardwarových kapacít.
V MySQL dátový typ majú aj premenné a parametre procedúr a funkcií.

Štandardné dátové typy v MySQL:

- Presné numerické
  - TINYINT, SMALLINT, MEDIUMINT , INT, BIGINT
  - DECIMAL (⇔ NUMERIC)

- Približné numerické
  - FLOAT, DOUBLE
- Reťazce
  - CHAR, VARCHAR
  - *BLOB, *TEXT
  - ENUM
    `ENUM('pondelok', 'utorok', 'streda', 'stvrtok', 'piatok')`
  - SET
    `SET('ja', 'ty', 'on', 'ona', 'ono') => 'on,ona' je dovolené`
- Dátum a čas
  - DATETIME, DATE, TIME
  - YEAR

Iné typy

  - XML, CURSOR, TABLE, NCHAR – SQL Server

Numerické typy

Typ	Min	Max	byte
TINYINT	-128	127	1
	0	255	
SMALLINT	-32768	32767	2
	0	655352	
MEDIUMINT	-8388604	8388607	3
	0	167772153	
INT	-2147483648	2147483647	4
	0	4294967295	
BIGINT	-9223372036854775808	9223372036854775807	8
	0	18446744073709551615	
FLOAT	±1.175494351E-38	±3.402823466E+384	4
DOUBLE	±2.2250738585072014E-308	±1.7976931348623157E+3088	8
DECIMAL(M,D)	1, 0	65, 30	M+2

```
USE DBmaz;
DROP TABLE IF EXISTS TabMaz;
CREATE TABLE IF NOT EXISTS TabMaz (i int, x SET('ja', 'ty', 'on') );
INSERT TabMaz VALUES(1,'Ja'), (2,'Ja,Ty');    ### 'Ja,Ty' bez medzery
```

DECIMAL(14,7) – desatinná aj celá časť má 7 cifier – potom 4, 4 bajty.
DECIMAL(18,8) – desatinná časť má 8 a celá časť 10 cifier – potom 4, 4+1 bajtov.

Počet cifier	Počet bajtov
0	0
1–2	1
3–4	2
5–6	3
7–9	4

```
use dbmaz;
DROP table IF EXISTS T;
CREATE TABLE T (x FLOAT, y DOUBLE, z DECIMAL(18, 9) );  # float 6+1, double 16+1 platnych cifier
INSERT INTO T VALUES (123456789.123456789, 123456789.123456789, 123456789.123456789);
SELECT * FROM T;   # '123457000', '123456789.12345679', '123456789.123456789'
```

Reťazcové typy

Štandardne 1 znak je 1 bajt, ale napr. č je dvojbajtový znak.

CHAR hodnoty pri uložení sú sprava doplnené s medzerami, ale pri načítaní tieto medzere sú **odstránené**.
VARCHAR hodnoty pri uložení **nie sú** doplnené s medzerami a medzery v hodnotách sú aj uložené aj načítané - presné.

```
DROP table xy;
CREATE TABLE xy (x VARCHAR(5), y CHAR(5));
INSERT INTO xy VALUES ('abčč  ', 'abčč  ');                  # Pozor, Pdf konvertuje 2 prázdne znaky z Word na 1
SELECT CONCAT(x, '|')x, CONCAT(y, '|') y FROM xy;            #  'abčč |', 'abčč|'
SELECT CHAR_LENGTH(x) nx, CHAR_LENGTH(y) ny FROM xy;         #  5, 4 - počet znakov
SELECT LENGTH(x) nx, LENGTH(y) ny FROM xy;                   #  7, 6  - počet bajtov  - č je dvojbajtový znak
```

Dátumové typy

Typ	Hodnota
DATETIME	'0000-00-00 00:00:00'
DATE	'0000-00-00'
TIME	'00:00:00'
YEAR	0

```
DROP table IF EXISTS T;
CREATE TABLE T (x datetime, y date );
INSERT INTO T VALUES (now(), CAST(now() AS DATE));
INSERT INTO T VALUES ('2015.01.02 00:00:02' , '2015-01-02 00:00:02');
INSERT INTO T VALUES ('2015-01-02' , '2015.01.02 00:00:02');
SELECT * FROM T;
```

x	y
2015-09-28 10:17:18	2015-09-28
2015-01-02 00:00:02	2015-01-02
2015-01-02 00:00:00	2015-01-02

Pretypovanie sa používa na konverziu hodnoty jedného typu na druhý typ. Rozoznávame

- **implicitné** pretypovanie – konverzia je typovo bezpečná a žiadne dáta sa nestratia. Napr. z menšieho na väčší celočíselný typ, alebo z float na reťazec.
- **explicitné** pretypovanie – je potrebné, keď môže dôjsť k strate informácie.

CAST(typ1 AS typ2)

Dovolené typy pre explicitnú konverziu (je ich málo)

BINARY[(N)]	CHAR[(N)]	DATE	DATETIME
DECIMAL[(M[,D])]	SIGNED [INTEGER]	TIME	UNSIGNED [INTEGER]

Poznamenáme, že float sa tu nenachádza (ide o implicitné).

Prevod z čísla na reťazec

```
SELECT CONCAT('x',1);          # 'x1' - impl. na string
SELECT CAST(Pi() AS CHAR);     # '3.141593' - expl. na string
```

Prevod z reťazca na číslo

```
SET @x = CAST( '3.141593' AS DECIMAL(4, 3) );
SELECT @x+1;   # 4.14200...
```

```
    # Počet cifier 12345678901234567890                    #   12345678901234567890
SELECT 5.0/3.0+0.0000000000;                               # 1.6666666667          – práve 10 cifier
SELECT 5.0/3.0+0.00000000000000000000;  # ⇔ :              # 1.66666666666666666600 – 18 cifier
SELECT CAST(5.0/3.0 as DECIMAL(21,20));                    # 1.66666666666666666600 – 18 cifier
SELECT  PI()  +  0.00000000000000000;                      # 3.14159265358979300       – 16 cifier
SELECT   CAST( 0.66666666666666666667 as DECIMAL(21,20)); # 0.66666666666666666667 – dec→dec
```

```
DROP TABLE IF EXISTS T;
CREATE TABLE IF NOT EXISTS T(x CHAR(8));
INSERT T VALUES('1.1'), ('2.1'), ('33.12345');
```

Explicitné pretypovanie

```
SELECT x, 10+CAST(x AS DECIMAL(4,2)) FROM T;        # ... // 43.12
### SELECT x, 10+CAST(x AS DECIMAL(3,2)) FROM T;  # POZOR!
```

1.1	11.10
2.1	12.10
33.12345	43.12

Implicit casting

```
SELECT x, x+10 FROM T;    # 43.12345 - na aký typ?
```

1.1	11.1
2.1	12.1
33.12345	43.12345

Reťazec vs DATATIME

```
DROP TABLE IF EXISTS T;
CREATE TABLE IF NOT EXISTS T(x DATETIME, y CHAR(19));
INSERT T VALUES(NOW(), '2015-01-01 00:00:01');
INSERT T VALUES(CAST(NOW() AS CHAR(19)), CAST('2015-01-01 00:00:02' AS DATETIME));
INSERT T VALUES('2015-01-01 00:00:03', NOW());
SELECT * FROM T;
```

x	y
2022-09-28 09:44:33	2015-01-01 00:00:01
2022-09-28 09:44:33	2015-01-01 00:00:02
2015-01-01 00:00:03	2022-09-28 09:44:33

3. Operátory

MySQL používa nasledujúce operátory:

a) Aritmetické
b) Operátor priradenia =
c) Logické
d) Bitové
e) Porovnávacie

a) Aritmetické

+, -, *, /, %, DIV

```
SELECT 7/3, 7/3 + 0.000000000, 7 DIV 3;    # DIV je celociselne delenie
 # 2.3333, 2.333333333, 2
```

c) Logické

AND, &&, OR, ||, XOR, NOT, !

SELECT NOT 3>1, NOT 0, NOT 1, NOT 2, (NOT 3) + 1, NOT 3 + 1, '<>', !3 + 1;

NOT 3>1	NOT 0	NOT 1	NOT 2	(NOT 3) + 1	NOT 3 + 1	<>	!3 + 1
0	1	0	0	1	0	<>	1

d) Bitové

|	OR
&	AND
~	Invert bits - NOT
Operátory posunu / shift	
<<	Posuň bity do ľava
>>	Posuň bity do prava

SELECT 10, BIN(10), HEX(10), HEX(16);

10	BIN(10)	HEX(10)	HEX(16)
10	1010	A	10

SELECT bin(2), bin(5), ~5;

bin(2)	bin(5)	~5
10	101	18446744073709551610

SELECT 5 &1, 5 & 2, 5 &3 ; ### 101 & 11

5 &1	5 & 2	5 &3
1	0	1

SELECT bin(~5), bin(~5 & 1), bin(~5|1);

bin(~5)	bin(~5 & 1)	bin(~5|1)
1111111111111111111111111111111111111111111111111111111111111010	0	1111111111111111111111111111111111111111111111111111111111111011

SELECT bin(~5), bin(2), bin(~5 & 2), ~5 & 2, 5 & 2, ~(5 & 2);

bin(~5)	bin(2)	bin(~5 & 2)	~5 & 2	5 & 2	~(5 & 2)
1111111111111111111111111111111111111111111111111111111111111010	10	10	2	0	18446744073709551615

SELECTbin(~5 | 2), ~5 | 2;

bin(~5 | 2)	~5 | 2
1111111111111111111111111111111111111111111111111111111111111010	18446744073709551610

5<<3 <=> 5*8

SELECT bin(5), bin(5<<3), 5<<3, '<=>', 5*8, bin(24), bin(24>>2), 24>>2, '<=>', 24 DIV 4 ; # 8=2^3

bin(5)	bin(5<<3)	5<<3	<=>	5*8	bin(24)	bin(24>>2)	24>>2	<=>	24 DIV 4
101	101000	40	<=>	40	11000	110	6	<=>	6

```
DROP TABLE IF EXISTS T;
CREATE TABLE IF NOT EXISTS T(i bigint);
INSERT T VALUES(2),(4);
INSERT T SET i = '000001000';
INSERT T SET i = b'000001000';      # pretypovanie na desatinne 8
##  INSERT T SET i = h'000001000';# NO
##  SELECT b'000001000'; # NO
INSERT T VALUES(b'000010000'),(3), (0x0000000F), (0x00000010);
SELECT i, bin(i), hex(i) FROM T;
```

i	bin(i)	hex(i)
2	10	2
4	100	4
1000	1111101000	3E8
8	1000	8
16	10000	10
3	11	3
15	1111	F
16	10000	10

e) Porovnávacie

=, >, >=, <, <=, <>, !=, !>, !<

Operátor	SELECT	Výsledok
>, <	1>2	0
<=, >=	2<=2	1
BETWEEN	3 BETWEEN 1 AND 10	1
NOT BETWEEN	3 NOT BETWEEN 1 AND 10	0
IN	5 IN (1,2,3,4)	0
NOT IN	5 NOT IN (1,2,3,4)	1
=	2=3	0
<>, !=	2<>3	1
<=>	NULL<=>NULL	1
LIKE	'Csaba' LIKE 'Cs%'	1
REGEXP	'Csaba' REGEXP '[Cc]s'	1
IS NULL	0 IS NULL	0
IS NOT NULL	0 IS NOT NULL	1

Priorita operátorov

Operátory vyššej priority sa vykonajú pred operátormi nižšej prior. http://dev.mysql.com/doc/refman/5.5/en/operator-precedence.html

- INTERVAL
- BINARY, COLLATE
- !
- - (unary minus), ~ (unary bit inversion)
- ^
- *, /, DIV, %, MOD
- -, +
- <<, >>
- &
- |
- = (comparison), <=>, >=, >, <=, <, <>, !=, IS, LIKE, REGEXP, IN
- BETWEEN, CASE, WHEN, THEN, ELSE

- NOT
- AND, &&
- XOR
- OR, ||
- = (assignment), :=

MySQL ponúka rôzne štandardné funkcie a operátory, ktoré sa najčastejšie používajú vo WHERE podmienke alebo v zozname za SELECT. http://dev.mysql.com/doc/refman/5.5/en/functions.html

```
SELECT [ ALL | DISTINCT ] [LIMIT ...]
        select_zoznam [ INTO tab3 ]  [ FROM zdroj_tab123]
[ WHERE podmienka_vyhlad_1]
[ GROUP BY group_zoznam ]
[ HAVING podmienka_vyhlad_2 ]
[ ORDER BY order_zoznam [ ASC | DESC ] ]
```

Nižšie uvedieme funkcie

- numerické
- reťazcové
- dátumu a času.

4. Numerické funkcie http://dev.mysql.com/doc/refman/5.1/en/numeric-functions.html

ABS()	ACOS()	ASIN()	ATAN()	ATAN2()	CEIL(), CEILING()	CONV()	COS()	COT()	CRC32()
DEGREES()	DIV	EXP()	FLOOR()	LN(), LOG()	LOG10()	LOG2()	%, MOD, MOD()	PI()	
POW(), POWER()	RADIANS()	RAND()	ROUND()	SIGN()	SIN()	SQRT()	TAN()	TRUNCATE()	

```
SELECT CEIL(3.14);        # 4
SELECT FLOOR(3.14);     # 3
SELECT ROUND(36.55, 1);  # 36.6
SELECT ROUND(36.55, 0); # 37
SELECT ROUND(36.55, -1);# 40
```

RAND() - je pseudonáhodná veličina z intervalu [0, 1).
Z intervalu [a; b): SELECT FLOOR(a + RAND() * (b − a));

```
SELECT FLOOR(5 + RAND() * (2));   ### 5 – 6 vykonaj viackrát
```

RAND(seed) - algoritmus generovania pseudonáhodných veličín sa štartuje vždy od seed

```
SELECT RAND(12), RAND(12), RAND(), RAND();  # 0.15741774081943347, 0.15741774081943347, …, …
SELECT 19 MOD 5;        # 4 ⇔
SELECT MOD(19, 5);      # 4 ⇔
SELECT 19 % 5;          # 4
SELECT EXP(1), LN(EXP(1)),  LOG(EXP(1));   # 2.718281828459045, 1, 1
SELECT PI()+0.00000000000000000;         # 3.14159265358979300
```

5. Reťazcové funkcie https://dev.mysql.com/doc/refman/5.1/en/string-functions.html

1) Tvorba reťazcov REPEAT, SUBSTRING, SUBSTRING_INDEX, LEFT, MID, RIGHT CONCAT, CONCAT_WS CHAR vs. ASCII	**3) Zmena reťazcov** REPLACE, INSERT REVERSE
2) Info o reťazcoch CHARACTER_LENGTH, LENGTH LOCATE, INSTR, STRCMP CHARSET	**4) Formátovanie reťazcov** LOWER, UPPER <=> LCASE, UCASE TRIM, LTRIM, RTRIM PAD, LPAD, RPAD

1) Tvorba reťazcov

Syntax

str1=REPEAT(str2, count)

str1=SUBSTRING(str2, position [, length]) <=>
str1=MID(str2, start [, length])

str1=LEFT(str2, length)
str1=RIGHT(str2, length)

Príklady

```
SELECT REPEAT('Ba', 2);                 #→'BaBa'  ⇔:
SET @x = REPEAT('Ba', 2);  SELECT @x;
SELECT SUBSTRING('Blava', 3, 2);        # 'av'
# <=>
SELECT MID('Blava', 3, 2);              # 'av'

SET @a=LEFT('Blava',2);          #'Bl'
SET @b=RIGHT('Blava',2);         #'va'
SELECT @a, @b;                   # 'Bl', 'va'
```

Vynechanie niektorých slov a znakov.

```
SELECT    SUBSTRING_INDEX('Dnes rano prsalo.', ' ',  1);  -- Dnes
SET @x = SUBSTRING_INDEX('Dnes rano prsalo.', ' ', -1); -- prsalo.
SELECT LEFT(@x, CHAR_LENGTH(@x) -1); -- prsalo (bez bodky)
```

Syntax

str1 = CONCAT(str2, str3, ...)
str1 = CONCAT_WS(delimiter, str2)

Príklady

```
SELECT CONCAT('Ta','try');              #'Tatry'
SELECT CONCAT_WS('-','Ta','try');       #'Ta-try'
```

Syntax
str1 = ASCII(str2)
string = CHAR(ascii code [,...])

Príklady
SELECT ASCII('A'); # 65

Vykonaj naraz:
```
SET @a=ASCII('Abba');  SET @b=ASCII('Baba');
SET @c=ASCII('abc');   SET @d=ASCII('b');      SET @e=ASCII('1');
SELECT @a,@b,@c,@d,@e;      # 65, 66, 97, 98, 49
```

```
DROP TABLE IF EXISTS T;
CREATE TABLE IF NOT EXISTS T(a CHAR(2));
INSERT T VALUES (CHAR(65)), (CHAR(66));
SELECT * FROM T;        # => A // B
```

2) Info o reťazcoch

Syntax
n = CHAR_LENGTH(string) ⇔ CHARACTER_LENGTH.
n = LENGTH(string) počet bajtov (pozri aj vyššie)
position = LOCATE(substring, string [, start_position])

Príklady
```
SET @i  = CHAR_LENGTH('Košice');      SELECT @i;  # 6
SET @cs= LENGTH('Košice');            SELECT @cs; # 7
```

```
SET @i1 = LOCATE('pa', 'Papasa', 1);     # 1
SET @i2 = LOCATE('pa', 'Papasa', @i1+1);
SELECT @i1, @i2;  # 1 // 3
```

Syntax
N = INSTR(str, substr)
N = STRCMP(str1, str2)

Príklady
```
SET @a = INSTR('Blava','av'); SELECT @a;        # 3
```

```
SET @a = STRCMP('Poprad','Presov');   # Po < Pr
SET @b = STRCMP('Poprad','Poprad');   # equal
SET @c = STRCMP('Presov', 'Poprad');
SELECT @a, @b, @c;                    # -1, 0, 1
```

3) Zmena reťazcov

Syntax

str1 = REPLACE(str2, search_string, replace_string)
string = INSERT(original_string, position, length, new_string)
str1 = REVERSE(str2)

Príklady

```
SET @a=REPLACE('C# and MySQL and R','and','a'); SELECT @a;  # 'C# a MySQL a R'
SET @a=INSERT('Blava',2,0,'ratis'); SELECT @a;  # 'Bratislava'
SET @a=INSERT('Blava',2,3,'ratis'); SELECT @a;  # 'Bratisa'
SET @a = REVERSE('indul a pap aludni'); SELECT @a;
```

4) Formátovanie reťazcov

Syntax

LOWER str1=LOWER(str2) <=> LCASE
UPPER str1=UPPER(str2) <=> UCASE

Príklady

```
SELECT LCASE('Blava, Poprad');   # blava, poprad
SELECT UCASE('Blava, Poprad');  # BLAVA, POPRAD
```

Syntax

Odstránenie medzier (alebo znakov)
TRIM([[BOTH|LEADING|TRAILING] [padding] FROM] str)
LTRIM(str)
RTRIM(str)

Doplnenie medzier
str1=LPAD(str2, length, pad)
str1=RPAD(str2, length, pad)

Príklady

```
SET @a=TRIM(LEADING '_' FROM '__Blava__');   # 'Blava__'
SET @b=TRIM('  Blava  ');                    # 'Blava'
SET @c=LTRIM('  Blava  ');                   # 'Blava  ';
SELECT CONCAT('|', @a, '|'), CONCAT('|', @b, '|'), CONCAT('|', @c, '|');
### =>  '|Blava__|', '|Blava|', '|Blava  |'

SELECT RPAD('Blava', 8, '.');  # Blava...
```

Iné

```
SELECT ELT(2, 'ja', 'ty', 'on');        # ty  –  2. prvok zoznamu
```

6. Dátumy a časové funkcie https://dev.mysql.com/doc/refman/5.1/en/date-and-time-functions.html

1) Zistenie dátumu a času CURDATE, NOW, CURTIME, SYSDATE	**3) Formátovanie dátumu a času** DATE_FORMAT, TIME_FORMAT
2) Extrahovanie častí dátumu a času DATE, TIME YEAR, QUARTER, MONTH, WEEK, DAY, HOUR, MINUTE, SECOND DAYNAME, MONTHNAME, lc_time_names DAYOFYEAR, DAYOFMONTH, DAYOFWEEK WEEKDAY, WEEKOFYEAR EXTRACT	**4) Sčítanie a odčítanie dátumu a času** DATE_ADD ⇔ ADDDATE DATE_SUB ⇔ SUBDATE ADDTIME SUBTIME
	5) Porovnanie dátumu a času DATEDIFF, TIMEDIFF, TIME_TO_SEC, SEC_TO_TIME

a) Zistenie dátumu a času

CURDATE( ), NOW( ) a CURTIME( ) vrátia čas v momente **vykonania príkazu**, v danom prípade SELECTu, a SYSDATE vrátia čas v momente **vykonania funkcie** SYSDATE:

```
SELECT CURDATE( ), NOW( ) , SLEEP(2), CURTIME( ), SLEEP(2), SYSDATE();
```

CURDATE( )	NOW( )	SLEEP(2)	CURTIME( )	SLEEP(2)	SYSDATE()
2015-09-30	2015-09-30 16:59:45	0	16:59:45	0	2015-09-30 16:59:49

b) Extrahovanie častí dátumu a času

Funkcie DATE a TIME vrátia iba dátum a iba čas.

```
SELECT NOW()  dt, DATE( NOW() )  d, TIME( NOW() ) t;
```

dt	d	t
2023-10-04 17:22:51	2023-10-04	17:22:51

Funkcie YEAR, MONTH, DAY, HOUR, MINUTE, SECOND extrahujú šesť komponentov DATE

```
SELECT NOW(), YEAR( NOW() )    r, MONTH( NOW() ) M, WEEK( NOW() ) T, DAY( NOW() )  d,
              HOUR( NOW() )  h, MINUTE( NOW() )m, SECOND( NOW())s;
```

NOW()	r	M	T	d	h	m	s
2017-10-03 17:59:50	2017	10	40	3	17	59	50

```
SET @d='1958-02-25';  ## ⇔  "1958-02-25"
SELECT   @d,  DAYOFYEAR(@d) r, DAYOFMONTH(@d) m, MONTHNAME(@d) nm,
              DAYNAME(@d) d1, DAYOFWEEK(@d) d2, WEEKDAY(@d) d3;
```

@d	r	m	nm	d1	d2	d3
1958-02-25	56	25	February	Tuesday	3	1

DAYOFWEEK používa číslovanie 1 = Sunday, 2 = Monday, ...,
kým WEEKDAY číslovanie 0 = Monday, 1 = Tuesday, ...

```
SET lc_time_names = 'sk_SK';  SELECT MONTHNAME(CURRENT_DATE()) slov;
SET lc_time_names = 'en_US'; SELECT MONTHNAME(CURRENT_DATE()) ang;
```

slov
február

Časť (**interval**) dátumu a času môžeme extrahovať aj pomocou funkcie EXTRACT pri pomoci zodpovedajúceho intervalu.

EXTRACT(*interval* FROM *date_time*) http://dev.mysql.com/doc/refman/5.0/en/date-and-time-functions.html#function_date-add

YEAR	yy	HOUR	hh
QUARTER	qq	MINUTE	mm
MONTH	mm	SECOND	ss
WEEK	ww	MICROSECOND	nn
DAY	dd	MINUTE_SECOND	mm:ss

Tab. Atomické intervaly

Poznamenáme, že existuje iba zložený interval MINUTE_SECOND, funkcia nie.

```
SELECT NOW(), EXTRACT( YEAR FROM NOW() )  r, EXTRACT( QUARTER FROM NOW() ) q,
              EXTRACT( MONTH FROM NOW() ) m , EXTRACT( WEEK FROM NOW() ) t,
              EXTRACT( DAY FROM NOW() ) d;
```

NOW()	r	q	m	t	d
2015-09-30 17:48:12	2015	3	9	39	30

```
SELECT NOW(), EXTRACT( HOUR FROM NOW() )  h, EXTRACT( MINUTE FROM NOW() ) m,
              EXTRACT( SECOND FROM NOW() )s, EXTRACT( MICROSECOND FROM NOW() ) ms;
```

NOW()	h	m	s	ms
2015-09-30 17:50:59	17	50	59	0

```
SELECT NOW(), EXTRACT( MINUTE_SECOND FROM NOW() ) m_s;
```

NOW()	m_s
2015-09-30 17:55:18	5518

c) Formátovanie dátumu a času

Pozrime sa na funkcie DATE_FORMAT a TIME_FORMAT, ktoré sa opierajú o špeciálne formátovacie kódy, ako napr. %a, %w, %W, %b, %M.

Code	Description	Results
%a	Abbreviated weekday name	(Sun…Sat)
%d	Day of the month (numeric)	(00…31)
%D	Day of the month with English suffix	(1st, 2nd, 3rd, etc.)
%e	Day of the month (numeric)	(0…31)
%j	Day of the year	(001…366)
%w, %W	Day of the week	(0=Sunday…6=Saturday)
%u	Week, where Monday is the first day of the week	(0…52)
%U	Week, where Sunday is the first day of the week	(0…52)
%v	Week, where Monday is the first day of the week; used with `%x'	(1…53)
%V	Week, where Sunday is the first day of the week; used with `%X'	(1…53)
%b	Abbreviated month name	(Jan…Dec)
%c	Month (numeric)	(1…12)
%m	Month (numeric)	(01…12)
%M	Month name	(January…December)
%y	Year (numeric, two digits)	(yy)
%Y	Year (numeric, four digits)	(yyyy)
%f	Microseconds (numeric)	(000000…999999)
%s	Seconds	(00…59)
%S	Seconds	(00…59)
%i	Minutes (numeric)	(00…59)
%h	Hour	(01…12)
%H	Hour	(00…23)
%I	Hour	(01…12)
%k	Hour	(0…23)
%l	Hour	(1…12)
%p	AM or PM	AM or PM
%r	Time, 12-hour	(hh:mm:ss [AP]M)
%T	Time, 24-hour	(hh:mm:ss)

Tab. Formátovacie kódy

```
SELECT CURDATE(), DATE_FORMAT( CURDATE(), ' %W, %M %e, %Y' ) d,
       CURTIME(), TIME_FORMAT( CURTIME(), ' %l: %i %p' ) t;
```

CURDATE()	d	CURTIME()	t
2015-09-30	Wednesday, September 30, 2015	19:37:01	7: 37 PM

d) Sčítanie a odčítanie dátumu a času

K dátumu môžeme pripočítať istý počet **intervalov**, ako(MONTH, YEAR, ...), pozri tab. vyššie, pomocou funkcií DATE_ADD a DATE_SUB, kde počet intervalov môže mať aj zápornú hodnotu s opačným efektom.

```
SELECT   NOW() n1,   DATE_SUB( NOW(), INTERVAL 2 MONTH) n2,  '=' eq,
                     DATE_ADD( NOW(), INTERVAL -2 MONTH) n2,
                     DATE_ADD( NOW(), INTERVAL  2 MONTH) n3,  '=' eq, NOW()+INTERVAL 2 MONTH n3;
```

n1	n2	eq	n2	n3	eq	n3
2015-09-30 20:20:58	2015-07-30 20:20:58	=	2015-07-30 20:20:58	2015-11-30 20:20:58	=	2015-11-30 20:20:58

Dni môžeme pripočítať (odčítať) k (od) dátumu aj pomocou funkcií ADDDATE, SUBDATE.

```
SELECT NOW() t, ADDDATE(NOW(), 10) t10, SUBDATE(NOW(), -10) t_10;
```

t	t10	t_10
2016-10-06 01:06:12	2016-10-16 01:06:12	2016-10-16 01:06:12

Čas pripočítame (odčítame) k (od) dátumu resp. času pomocou funkcií ADDTIME, SUBTIME.

```
SET @d=NOW();
SELECT @d,  ADDTIME(@d, '01:00:00') dt1,  SUBTIME(@d, '01:00:00') dt2;
```

@d	dt1	dt2
2015-09-30 21:24:47	2015-09-30 22:24:47	2015-09-30 20:24:47

e) Porovnanie dátumu a času

DATEDIFF, TIMEDIFF, TIME_TO_SEC, SEC_TO_TIME
Porovnanie dátumu (v dňoch) a času sa štandardne uskutočňuje pomocou DATEDIFF, TIMEDIFF.

```
SELECT CURDATE()d2, '2015,9,1' d1, DATEDIFF(CURDATE(), DATE('2015,09,01') ) rozd;  # <=>:
SELECT CURDATE()d2, '2015,9,1' d1, DATEDIFF(CURDATE(), '2015,9,1' ) rozd;
```

d2	d1	rozd
2015-09-30	2015,9,1	29

```
SELECT CURTIME()t2, '02:30:00' t1, TIMEDIFF(CURTIME(), TIME('02:30:00') ) rozd;  # <=>:
SELECT CURTIME()t2, '02:30:00' t1, TIMEDIFF(CURTIME(), '02:30:00' ) rozd;
```

t2	t1	rozd
19:14:30	02:30:00	16:44:30

Porovnanie času môžeme uskutočniť aj na báze sekúnd po transformovaní času na sekundy pomocou funkcie TIME_TO_SEC. Opačnou funkciou je SEC_TO_TIME.

```
SELECT TIME('02:30:00') t, TIME_TO_SEC('02:30:00') s,
       TIME_TO_SEC('02:30:00')/60 as m, TIME_TO_SEC('02:30:00')/60/60 as h;
```

t	s	m	h
02:30:00	9000	150.0000	2.50000000

```
set @s = TIME_TO_SEC('02:30:00');
SELECT SEC_TO_TIME(@s) s;       # '02:30:00'

select '2012-11-19' > '2012-11-18';                 # 1
select date('2012-11-19') > date('2012-11-18'); # 1
```