2. DATABÁZE

2.1. Výpis databází MySQL

SHOW DATABASES;

- tento dotaz zobrazuje jména databází ve spuštěném MySQL

2.2. Založení databáze

CREATE DATABASE nazev\_databaze;

- tento příkaz vytvoří databázi se jménem "nazev\_databaze" (délka názvu může být max. 65 znaků)

- abychom mohli databázi používat musíme v ní vytvořit jednu či více tabulek příkazem CREATE TABLE (viz níže)

- při práci v příkazovém řádku musíme určit aktivní databázi příkazem USE (viz níže)

2.3. Nastavení aktivní databáze

USE nazev\_databaze;

- databázi "nazev\_databaze" nastavíme takto jako aktivní a můžeme s ní pracovat

2.4. Název aktuální databáze

SELECT DATABASE();

- vrací název aktuální databáze

2.5. Výpis seznamu tabulek v databázi

SHOW TABLES;

- zobrazí seznam tabulek aktuální databáze

SHOW TABLES FROM nazev\_databaze;

- zobrazí seznam tabulek z databáze "nazev\_databaze"

SHOW TABLE STATUS FROM nazev\_databaze;

- zobrazí souhrnný seznam informací o jednotlivých tabulkách databáze, př.: Avg\_row\_length, Data\_length, Max\_data\_length, Index\_length...

2.6. Zadávání příkazů ze souboru

SOURCE cesta/soubor;

- př.: SOURCE /moje/prikazy/zal\_knih.mysql;

- MySQL vykoná všechny příkazy uvedené v souboru "zal\_knih.mysql"

2.7. Smazání databáze

DROP DATABASE nazev\_databaze;

- vymaže celou databázi se jménem "nazev\_databaze", tedy všechny tabulky a data v nich uložená

Úvod | Databáze | Tabulky | Datové typy | Práce s daty | Přílohy

3. TABULKY

3.1. Vytvoření tabulky

CREATE TABLE nazev\_tabulky (nazev\_sloupce datovy\_typ,... );

- v databázi, která je právě aktivní vytvoříme novou tabulku (typu MYISAM)

- délka názvu tabulky (a sloupců) může být max. 65 znaků

- příklad:

CREATE TABLE knihovna (autor VARCHAR(20),

kniha VARCHAR(20) NOT NULL PRIMARY KEY,

stran SMALLINT UNSIGNED,

rok YEAR(4),

poznamka ENUM('neprecteno','precteno','pujceno') DEFAULT 'neprecteno');

- datové typy jsou popsány v samostatné kapitole viz: datové typy

3.2. Typy tabulek

CREATE TABLE nazev\_tabulky (nazev\_sloupce datovy\_typ,... ) 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

- před nasazením jiného typu než MYISAM si prostudujte originální dokumentaci

3.3. Vytvoření dočasné tabulky

CREATE TEMPORARY TABLE nazev\_tabulky (nazev\_sloupce datovy\_typ,... );

- takto vytvoříme dočasnou, která po uzavření spojení s databází zanikne

3.4. Výpis popisu tabulky

DESCRIBE nazev\_tabulky;

SHOW COLUMNS FROM nazev\_tabulky;

- tento dotaz nám zobrazí definici požadované tabulky (názvy + datové typy + modifikátory)

3.5. Změny v tabulce

ALTER TABLE nazev\_tabulky prikaz1, prikaz2, prikaz3, prik...;

- provede nějaký příkaz/příkazy s tabulkou "nazev\_tabulky", viz dále:

Nový sloupec

.. ADD nazev\_noveho\_sloupce datovy\_typ;

.. ADD COLUMN nazev\_noveho\_sloupce datovy\_typ;

- příkaz přidá do tabulky nový sloupec

- př.: ALTER TABLE knihovna ADD COLUMN vydavatel VARCHAR(10);

- modifikátory:

.. FIRST

- přidá nový sloupec na začátek tabulky

- př.: ALTER TABLE knihovna ADD COLUMN cislo SMALLINT FIRST;

.. AFTER nazev\_sloupce;

- přidá nový sloupec za sloupec "nazev\_sloupce"

- př.: ALTER TABLE knihovna ADD COLUMN zanr VARCHAR(10) AFTER kniha;

Smazání sloupce

.. DROP nazev\_odstranovaneho\_sloupce;

.. DROP COLUMN nazev\_odstranovaneho\_sloupce;

- příkaz odebere požadovaný sloupec

- př.: ALTER TABLE knihovna DROP vydavatel;

Změna parametrů

.. CHANGE nazev\_sloupce novy\_nazev\_sloupce nove\_nastaveni;

- změní datový typ a může sloupec i přejmenovat

- př.: ALTER TABLE knihovna CHANGE kniha knihy VARCHAR(30) NOT NULL;

Modifikace parametrů

.. MODIFY nazev\_sloupce nove\_nastaveni;

- u požadovaného sloupce změní datový typ

- př.: ALTER TABLE knihovna MODIFY kniha VARCHAR(30) NOT NULL;

Přejmenování tabulky

.. RENAME novy\_nazev\_tabulky;

- příkaz přejmenuje požadovanou tabulku

- př.: ALTER TABLE knihovna RENAME knihovnicka;

3.6. Indexy a klíče v tabulkách

SHOW KEYS FROM nazev\_tabulky;

SHOW INDEX FROM nazev\_tabulky;

- vypíše podrobné informace o primárních klíčích a indexech v tabulce

3.7. 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

3.8. Smazání tabulky

DROP TABLE nazev\_tabulky;

- odstraní z aktivní databáze tabulku s názvem "nazev\_tabulky"

3.9. Optimalizace tabulky

OPTIMIZE TABLE nazev\_tabulky;

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

Úvod | Databáze | Tabulky | Datové typy | Práce s daty | Přílohy

4. DATOVÉ TYPY

4.1. Základní informace

- pro omezení délky řetězce (maximální velikost je 255) používáme parametr "m", zápis je: datovy\_typ(m)

- př.: TINYINT(1), nebo VARCHAR(100)

- u reálných čísel používáme navíc parametr "d" (maximální velikost je 30)

- tímto parametrem omezíme délku čísla za desetinou čárkou, zápis je: datovy\_typ(m,d)

- př.: FLOAT(5,3)

- sloupce určené jako INDEXY (nebo i PRIMARY KEY) označíme na konci deklarace tabulky:

- př.: CREATE TABLE pokus (jm CHAR(20) NOT NULL, cis INT, PRIMARY KEY (jm), INDEX(cis));

- název indexu (INDEX nazev (sloupec)) zadáváme pokud bude indexů více

4.2. Celá čísla

TINYINT

- 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)

MEDIUMINT

- rozsah hodnot od -8 388 608 do +8 388 607, bez znaménka 0 až 16 777 215 (zabere 3 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)

BIT nebo BOOL

- synonymum pro TINYINT(1)

4.3. 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)

- rozsah nastavíme parametry "m" (počet číslic celkem) a "d" (počet desetinných míst), maximální rozsah je stejný s typem DOUBLE

DEC(m,d) nebo NUMERIC(m,d)

- synonyma pro typ DECIMAL(m,d)

4.4. Datum a čas

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 do buňky tohoto typu 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) bude rozsah 1901 až 2155, formát je "RRRR", při YEAR(2) bude rozsah 1970-2069

4.5. Řetězce

CHAR(m)

- délka řetězce "m" může být v rozsahu 0-255

- 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

4.6. Modifikátory

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 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

DEFAULT vychozi\_hodnota

- pokud bude buňka prázdná, systém do ní automaticky přiřadí hodnotu "vychozi\_hodnota"

- řetězce nezapomeňte psát v uvozovkách

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 píkazů 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

- 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

UNIQUE

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

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

- př.: TINYINT má rozsah -118 až +127 a TINYINT UNSIGNED má rozsah 0 až 255

ZEROFILL

- použití u čísel, dotaz 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

Úvod | Databáze | Tabulky | Datové typy | Práce s daty | Přílohy

5. PRÁCE S DATY

5.1. Vkládání záznamů

INSERT INTO nazev\_tabulky VALUES (seznam\_hodnot);

- pro všechny sloupce v tabulce "nazev\_tabulky" musíme vložit data

- př.: INSERT INTO knihovna VALUES ('Oranžový Oto','Tropické ovoce',110,2003,'neprecteno');

- nebo jen do některých sloupců

- př.: INSERT INTO knihovna (autor,kniha) VALUES ('Oranžový Oto','Tropické ovoce');

5.2. Vkládání záznamů ze souboru

LOAD DATA LOCAL INFILE 'jmeno\_souboru' INTO TABLE nazev\_tabulky;

- příkaz vloží do tabulky "nazev\_tabulky" data ze souboru "jmeno\_souboru", který je lokálně uložen na PC

- př.: LOAD DATA LOCAL INFILE 'nove\_knihy.txt' INTO TABLE knihovna FIELDS TERMINATED BY ',' ENCLOSED BY '"' LINES TERMINATED BY '\n';

- záznamy jsou v uvozovkách, oddělené čárkou a konce řádků máme zakončené odentrováním

- pokud je pořadí sloupců v souboru odlišné, musíme je připsat do závorky za název tabulky

- modifikátory:

.. FIELDS TERMINATED BY 'neco'

- znak oddělující jednotlivé záznamy, většinou čárka ',' nebo tabulátor '\t'

.. ENCLOSED BY 'neco'

- znak uzavírající hodnoty záznamů, většinou uvozovky '"'

.. LINES TERMINATED BY 'neco'

- znak ukončující řádky, většinou odentrování '\n'

.. LOW\_PRIORITY

- př.: LOAD DATA LOW\_PRIORITY LOCAL INFILE...

- MySQL uloží data do tabulky až se s ní přestanou všichni pracovat

5.3. Obnova záznamů

UPDATE nazev\_tabulky SET jmeno\_sloupce=nova\_hodnota WHERE podminka;

- př.: UPDATE knihovna SET stran='260' WHERE kniha='Lesnictví';

- u knihy "Lesnictví" jsme upravili počet stran

5.4. Výpis záznamů

SELECT pozadavky FROM podminky\_vyberu;

SELECT pozadavky FROM podminky\_vyberu1 UNION SELECT pozadavky FROM podminky\_vyberu2;

- př.: SELECT autor FROM knihovna;

- tento dotaz nám vytáhne z tabulky "knihovna" všechny autory

- př.: SELECT autor,kniha FROM knihovna;

- tento dotaz nám vytáhne z tabulky "knihovna" všechny autory a knihy

- př.: SELECT kniha FROM knihovna UNION SELECT kniha FROM knihovna2;

- pomocí UNION můžeme spojit výběr z dvou tabulek ("pozadavky" musí být shodné); zavedeno v MySQL 4+

- pomocí UNION ALL budou výstupem i opakující se hodnoty, které UNION standardně nevrací

- př.: SELECT (2\*5/3)+4;

- i tohle funguje!

- př.: SELECT BENCHMARK(1000000,1+1);

- jednoduchý výkonnostní test rychlosti DB serveru

- př.: EXPLAIN SELECT pozadavky FROM podminky\_vyberu;

- příkazem EXPLAIN získáme informace o tom, jak MySQL dotaz SELECT provede

- seznam příkazů a podmínek následuje:

Vyber vše

.. \*

- př.: SELECT \* FROM knihovna;

- hvězdička nám vytáhne z tabulky "knihovna" všechna data

Výběr části dat podle podmínky

.. WHERE podminka;

- př.: SELECT \* FROM knihovna WHERE poznamka='precteno';

- vytáhne všechny informace o knihách které jsou přečtené "precteno"

- př.: SELECT kniha FROM knihovna WHERE poznamka='precteno';

- vytáhne názvy knih, které jsou označeny jako přečtené "precteno"

- př.: SELECT knihovna.kniha FROM knihovna,cetba WHERE knihovna.kniha=cetba.kniha;

- tabulku "knihovna" už známe, zde je navíc tabulka "cetba", která obsahuje informace o přečtených knihách

- příklad nám vytáhne názvy knih z knihovny ("knihovna"), které máme v knihovně ("knihovna") a četli jsme je ("cetba")

Porovnávací operátory

.. = a další...

- = (rovno), <> (nerovno), < (menší), <= (menší nebo rovno), > (větší), >= (větší nebo rovno)

- <=> (rovno; včetně hodnot NULL), != (nerovno; stejné jako <>)

.. x BETWEEN x1 AND x2;

- určí zda se "x" nachází mezi hodnotami "x1" až "x2" (včetně těchto hodnot)

- př.: SELECT \* FROM knihovna WHERE rok BETWEEN 1990 AND 2000;

- takto vypíšeme informace o knihách z knihovny, které vyšli mezi roky 1990 (včetně) a 2000 (včetně)

.. x NOT BETWEEN x1 AND x2;

- určí zda "x" je mimo hodnoty "x1" až "x2" (včetně těchto hodnot); je to tedy opak k operátoru BETWEEN

.. IN (kde\_hledat)

- hledá hodnoty dle zadaného seznamu

- př.: SELECT kniha FROM knihovna WHERE rok IN(2001,2002,2003);

- MySQL vypíše knihy z let 2001-2003

- ! v závorce může být i standardní dotaz: SELECT neco FROM tabulka WHERE podminka;

.. NOT IN

- opak IN

.. IS NULL;

- nulová hodnota

- př.: SELECT kniha FROM knihovna WHERE stran IS NULL;

- takto zjistíme knihy s nevyplněným políčkem počet stran

.. IS NOT NULL

- opak nulové hodnoty

.. LIKE

- upřesnění výběru

- př.: SELECT kniha FROM knihovna WHERE autor LIKE 'Z%';

- operátor LIKE vybere knihy jejichž autor začíná od Z

- procento "%" nahrazuje libovolný počet znaků, podtržítko "\_" pouze jeden znak

.. NOT LIKE

- opak k operátoru LIKE

Regulární výrazy

.. x REGEXP r;

.. x RLIKE r;

- výsledkem je pravda - pokud hodnota "x" odpovídá regulární hodnotě "r"

.. x NOT REGEXP r;

.. x NOT RLIKE r;

- výsledkem je pravda - pokud hodnota "x" NEodpovídá regulární hodnotě "r"

- přehled symboliky regulárních výrazů:

- x\* - počet výskytu "x" je 0 nebo více

(- xy\* - počet výskytu "y" je 0 nebo více)

(- xyz\* - počet výskytu "z" je 0 nebo více)

(- (xyz)\* - počet výskytu "xyz" je 0 nebo více)

- x? - počet výskytu "x" je 0 nebo 1

- x+ - počet výskytu "x" je 1 nebo více

- x{n} - počet výskytu "x" je n

- x{n,m} - počet výskytu "x" je n až m

- x{n,} - počet výskytu "x" je n nebo více

- ^x - řetězec začíná "x"

- x$ - řetězec končí "x"

- . - jakýkoliv jeden znak

- [a-z] - jakýkoliv znak mezi "a" až "z"

- [0-9] - číslo

- [abcd123] - jakýkoliv znak ze závorky

- | - slouží pro oddělení řetězců ve výrazu

- a teď několik příkladů:

- př.: SELECT 'abcdefg' REGEXP 'cde';

- př.: SELECT 'abcdefg' LIKE '%cde%';

- totožné příklady jejichž výsledkem je 1 (pravda)

- př.: SELECT 'abcdefg' REGEXP '^cde$';

- př.: SELECT 'abcdefg' LIKE 'cde';

- totožné příklady jejichž výsledkem je 0 (nepravda)

- př.: SELECT 'abcdefg' REGEXP 'bc|ef';

- př.: SELECT 'abcdefg' REGEXP 'ef|bc';

- př.: SELECT 'abcdefg' LIKE '%bc%ef%';

- totožné příklady jejichž výsledkem je 1 (pravda)

Výběr dat funkcí JOIN

.. tabulka1 LEFT JOIN tabulka2 podminka;

.. tabulka2 RIGHT JOIN tabulka1 podminka;

- př.: SELECT \* FROM tabulka1,tabulka2 WHERE tabulka1.id=tabulka2.id;

- př.: SELECT \* FROM tabulka1 LEFT JOIN tabulka2 ON tabulka1.id=tabulka2.id;

- oba předchozí příklady vykonávájí skoro to samé, ale funkce JOIN vrátí odpovídající řádky levé tabulky (tabulka1) bez ohledu na to, zda k těmto řádkům existuje nějaký odpovídající řádek v druhé tabulce (tabulka2)

- funkce WHERE totiž nevrácí výsledek tam kde jsou nulové hodnoty (NULL)

- př.: SELECT \* FROM tabulka1 LEFT JOIN tabulka2 USING (id);

- zkrácený zápis předchozího příkladu

Pojmenování

.. jmeno AS nove\_jmeno

- př.: SELECT k.kniha FROM knihovna AS k, cetba AS c WHERE k.kniha=c.kniha;

- pomocí klíčového slova AS můžeme pojmenováním zkrátit zápis předchozího příkladu

- př.: SELECT autor,kniha,(cena\*0.95) AS 'cena\_bez\_dph' FROM knihovna;

- pokud bychom měli u knih i cenu (sloupec "cena"), takto si ji necháme vypsat knihy a cenu bez DPH

- př.: SELECT 'Jmeno:' AS 'jmeno\_autora', autor FROM knihovna;

- zde nám MySQL vypíše vedle jmen autorů sloupec s názvem "jmeno\_autora" s DEFAULT hodnotou "Jmeno:"

Spojení proměnných

.. CONCAT(promenne\_pro\_spojeni)

- př.: SELECT CONCAT(kniha,' - ',autor) AS knihautor FROM knihovna;

- vypíše novy sloupec "knihautor", který bude obsahovat data ve formátu: název knihy - název autora

.. CONCAT\_WS(slucovac,promenne)

- př.: SELECT CONCAT\_WS('.','www','junext','net');

- vypíše: www.junext.net

Odstranění duplikátů

.. DISTINCT

- př.: SELECT DISTINCT poznamka FROM knihovna;

- tento příklad nám vypíše jaké používáme poznámky, tedy P,N,U (bez DISTINCT by vypsal vše: P,N,U,P,P,N)

Slučování do skupin

.. GROUP BY

- př.: SELECT poznamka, SUM(stran) AS 'celkem\_stran' FROM knihovna GROUP BY poznamka;

- sečte (příkaz SUM) počet stran u knih seskupených dle poznámek (P-přečteno, N-nepřečteno...)

Omezení počtu

.. LIMIT start,pocet;

- př.: SELECT kniha FROM knihovna WHERE poznamka='neprecteno' LIMIT 0,5;

- najde názvy prvních 5 knih, které jsou v poznámce označeny jako nepřečtené

- v tomto případě lze použít i zápis: ...LIMIT 5;

Seřazení

.. ORDER BY podminka;

- př.: SELECT \* FROM knihovna ORDER BY autor,kniha;

- vybere z tabulky všechny informace a srovná je vzestupně podle jmen autorů a názvů knih

.. ORDER BY podminka DESC;

- př.: SELECT \* FROM knihovna ORDER BY autor DESC;

- srovná výpis podle autorů, tentokrát sestupně

.. ORDER BY RAND();

.. ORDER BY RAND(N);

- př.: SELECT kniha FROM knihovna WHERE poznamka='neprecteno' ORDER BY RAND() LIMIT 1;

- výstupem je jedna nepřečtená kniha náhodně vybraná

- zadáním parametru "N" určíme výchozí hodnotu pro výpočet náhodného čísla

Logické operátory

- výstupem jsou nalezené hodnoty, popřípadě pravdivostní hodnota: "1","true" (pravda) nebo "0","false" (nepravda)

.. AND, &&

- př.: SELECT kniha FROM knihovna WHERE poznamka='neprecteno' AND rok<2000;

- AND nám zde vybere nepřečtené knihy vydané před rokem 2000

.. OR, ||

- př.: SELECT kniha FROM knihovna WHERE poznamka='neprecteno' || poznamka='precteno';

- výstupem jsou všechny nepřečtené a přečtené knihy

.. NOT, !

- negace dotazu např. SELECT NOT(1 AND 1); zde je výsledkem 0

Kontrolní funkce

.. CASE ... END;

- př.: SELECT CASE hledana\_hodnota WHEN 1 THEN 'jedna' WHEN 2 THEN 'dva' ELSE 'tri a vice' END;

- pokud hledana\_hodnota bude 1 vypíše MySQL "jedna", pokud 2 vypíše "dva", v ostatních případech "tri a vice"

.. IF(podminka,pravda,nepravda);

- př.: SELECT IF(10>9,'ano','ne');

- vypíše "ano"

.. IFNULL(podminka,vystup\_pri\_chybe);

- př.: SELECT IFNULL(1/0,'chyba');

- dělíme nulou což je blbost, tak to vypíše "chyba"

.. NULLIF(promenna1,promenna2);

- vrací promenna1, pokud se promenna1 nerovná promenna2 (v opačném případě vrácí NULL)

Aritmetické operátory

- přehled operátorů: + (součet), - (odečet), \* (součin), / (podíl), % (zbytek po podílu)

- př.: SELECT 8%3;

- výsledkem je hodnota 2

Manipulace s čísly (agregační fce)

.. AVG(nazev\_sloupce)

- spočítá průměr numerických hodnot ve sloupci

- př.: SELECT AVG(stran) FROM knihovna;

.. COUNT(nazev\_sloupce)

- spočítá počet hodnot ve sloupci

.. COUNT(DISTINCT nazev\_sloupce)

- spočítá počet jedinečných hodnot ve sloupci

.. GREATEST(hodnota1,hodnota2,hodno...)

- př.: SELECT GREATEST(10,3,7,24);

- vrátí největší hodnotu (24)

- funkce funguje i pro text (při zadání "J","U","N" vrátí U)

.. LEAST(hodnota1,hodnota2,hodno...)

- př.: SELECT LEAST(10,3,7,24);

- vrátí nejmenší hodnotu (3)

- funkce funguje i pro text (při zadání "J","U","N" vrátí J)

.. MAX(nazev\_sloupce)

- př.: SELECT kniha, MAX(stran) FROM knihovna;

- dotaz nám najde knihu s nejvyšším počtem stran

.. MIN(nazev\_sloupce)

- opak MAX(nazev\_sloupce)

.. MOD(delenec,delitel)

- vyplivne zbytek po dělení

.. ROUND(cislo)

- zaokrouhlí zadané "cislo" na celé číslo

.. ROUND(cislo,pocet\_mist)

- zaokrouhlí "cislo" na zadaný počet desetiných

.. STD(nazev\_sloupce)

- spočítá směrodatnou odchylku číselných hodnot ve sloupci

.. SUM(nazev\_sloupce)

- provede součet číselných hodnot ve sloupci

Manipulace s textem

.. LENGTH(retezec);

- př.: SELECT LENGTH('abeceda');

- funkce vrací délku řetězce; v tomto případě je to 7

.. LOCATE(co\_hledat,v\_cem,kde\_zacit);

- př.: SELECT LOCATE('ce','abeceda',1);

- hledá řetězec "ce" v řetězci "abeceda" od pozice 1; výsledkem je 4

.. SUBSTRING(retezec,kde\_zacit);

- př.: SELECT SUBSTRING('abeceda',4);

- vypíše řetězec od zadané pozice, tedy "ceda"

.. REPLACE(retezec,co\_nahradit,cim\_nahradit);

- př.: SELECT REPLACE('abeceda','abec','nezb');

- nahrazuje části řetězce; vypíše "nezbeda"

.. REVERSE(retezec);

- př.: SELECT REVERSE('abeceda');

- otáčí řetězce; vypíše "adeceba"

.. TRIM(retezec);

- př.: SELECT TRIM(' abeceda ');

- oseká řetězec o mezery a vypíše "abeceda"

.. TRIM(BOTH retezec1 FROM retezec2);

- př.: SELECT TRIM(BOTH 'a' FROM 'abeceda');

- vypíše "beced"

.. TRIM(LEADING retezec1 FROM retezec2);

- př.: SELECT TRIM(LEADING 'a' FROM 'abeceda');

- vypíše "beceda"

.. TRIM(TRAILING retezec1 FROM retezec2);

- př.: SELECT TRIM(TRAILING 'a' FROM 'abeceda');

- vypíše "abeced"

.. LTRIM(retezec);

- př.: SELECT LTRIM(' abeceda ');

- vypíše "abeceda "

.. RTRIM(retezec);

- př.: SELECT RTRIM(' abeceda ');

- vypíše " abeceda"

.. UPPER(retezec);

.. LOWER(retezec);

- př.: SELECT UPPER('abeceda');

- vypíše "ABECEDA"

- UPPER převádí písmena zadaného řetězce na velká, LOWER na malá

Manipulace s datumem a časem

SELECT NOW();

- dotaz vrátí aktuální datum a čas ve tvaru RRRR-MM-DD HH:MM:SS

- modifikace SELECT NOW()+0; vrátí tvar RRRRMMDDHHMMSS

SELECT CURRENT\_DATE();

- aktuální datum (RRRR-MM-DD)

SELECT CURRENT\_TIME();

- aktuální čas (HH:MM:SS)

SELECT DATE\_FORMAT(vstup,vystup);

- př.: SELECT DATE\_FORMAT(NOW(),"%w.%e.%y");

- %Y - rok RRRR (př. 2003, 1999 ...)

- %y - rok RR (př. 03, 99 ...)

- %m - měsíc MM (př. 01, 06, 12 ...)

- %c - měsíc M nebo MM (př. 1, 6, 12 ...)

- %M - název měsíce (př. January ...)

- %b - název měsíce zkráceně (př. Jan, Feb ...)

- %u - číslo týdne v roce - %D - den řadovou číslovkou (př. 1st, 2nd ...)

- %d - den v měsíci DD (př. 01, 02, 31 ...)

- %e - den v měsíci D nebo DD (př. 1, 2, 31 ...)

- %w - číslo dne v týdnu D (př. 0, 6 ...)

- %W - název dne v týdnu (př. Sunday ...)

- %a - název dne v týdnu zkráceně (př. Sun, Mon ...)

- %j - číslo dne v roce DDD (př. 000, 006, 366 ...)

- %H - hodina HH (př. 00, 06, 23 ...)

- %k - hodina H nebo HH (př. 0, 6, 23 ...)

- %h - hodina HH jen do 12 (př. 01, 06, 12 ...)

- %l - hodina H nebo HH jen do 12 (př. 1, 6, 12 ...)

- %i - minuty MM (př. 01, 06, 59 ...)

- %s - sekundy SS (př. 01, 06, 59 ...)

- %P - délka cyklu - půldenní nebo celodenní (př. AM, PM)

- %% - znak %

SELECT QUARTER(datum);

- vrací číslovku čtvrtletí dle zadaného data (RRRR-MM-DD)

Atd...

- v originální dokumentaci MySQL ( http://dev.mysql.com/doc/refman/5.1/en/functions.html) jsou ještě další funkce

- pokud jste zde nenašli co potřebujete - laskavě se tam podívejte

Uživatelské proměnné

SET @a=hodnota;

SELECT @a:=hodnota;

- do proměnné "a" se uloží nějaká "hodnota", kterou si MySQL pamatuje do konce aktuálního spojení

- (proměnnou nelze zatím použít úplně ve všech dotazech MySQL)

- př.: SET @a='precteno'; SELECT \* FROM knihovna WHERE poznamka=@a;

5.5. Výpis záznamů do souboru

SELECT \* INTO OUTFILE 'nazev\_vystupniho\_souboru' FIELDS TERMINATED BY ';' FROM nazev\_tabulky;

- příkaz zapíše data z tabulky "nazev\_tabulky" do souboru a jednotlivé položky oddělí středníkem

- př.: SELECT \* INTO OUTFILE 'prectene.txt' FIELDS TERMINATED BY ',' FROM knihovna WHERE poznamka='precteno';

- příklad zapíše do souboru informace o přečtených knihách a oddělí je čárkou

5.6. Mazání záznamů

DELETE FROM nazev\_tabulky WHERE podminka;

- př.: DELETE FROM knihovna WHERE kniha='Horníkův den';

- příkaz nám vymaže knihu "Horníkův den" z tabulky, tedy celý řádek

DELETE FROM nazev\_tabulky;

- příkaz nám vymaže všechny záznamy v tabulce

TRUNCATE nazev\_tabulky;

- dělá to samé jako předešlý příkaz, ale je rychlejší (smaže tabulku a zase jí založí)

Úvod | Databáze | Tabulky | Datové typy | Práce s daty | Přílohy

6. PŘÍLOHY

6.1. Ukázka jednoduché MySQL databáze

- v níže uvedeném schématu máme například založeny dvě databáze, v první jsou dvě tabulky a v druhé je jedna tabulka

- samotná data jsou pak uvnitř jednotlivých tabulek

- a jak například vypadá tabulka s daty, vidíte níže v příkladu tabulky "knihovna"

MySQL

+-------------------------------------------------+

| |

| Databáze1 Databáze2 |

| +--------------------------+ +-------------+ |

| | | | | |

| | Tabulka1 Tabulka2 | | Tabulka1 | |

| | |--------| |--------| | | |--------| | |

| | | Data | | Data | | | | Data | | |

| | |--------| |--------| | | |--------| | |

| | | | | |

| +--------------------------+ +-------------+ |

| |

+-------------------------------------------------+

6.2. Příklad tabulky "knihovna"

- jako primární klíč jsem zvolil název knihy, protože nepředpokládám, že by se opakoval

knihovna

+--------------------------------------------------------------------------+

| autor kniha stran rok poznamka |

| VARCHAR(20) VARCHAR(20) SMALLINT YEAR(4) SET ('neprecteno', |

| NOT NULL UNSIGNED 'precteno','pujceno') |

| PRIMARY KEY DEFAULT 'neprecteno' |

| +--------------+---------------+---------+--------+------------------+ |

| | Bílý Josef | Malujeme byt | 129 | 2001 | precteno | |

| |--------------|---------------|---------|--------|------------------| |

| | Černý Tomáš | Horníkův den | 96 | 1985 | neprecteno | |

| |--------------|---------------|---------|--------|------------------| |

| | Červený Jiří | Asertivita | 198 | 1996 | precteno,pujceno | |

| |--------------|---------------|---------|--------|------------------| |

| | Zelený Karel | Lesnictví | 250 | 1999 | precteno | |

| |--------------|---------------|---------|--------|------------------| |

| | Zelený Petr | Alkohol a my | 203 | 2001 | precteno | |

| |--------------|---------------|---------|--------|------------------| |

| | Žlutý Standa | Historie Číny | 367 | 2003 | neprecteno | |

| +--------------+---------------+---------+--------+------------------+ |

+--------------------------------------------------------------------------+