

# Databáze

Databázový server MariaDB



# Co je to databáze?

**Databáze je uspořádaná množina dat z reálného světa,  
uložená na paměťovém médiu.**

V širším smyslu se databází myslí i **aplikace** (software), umožňující manipulaci s daty.

Tento software se nazývá **Systém Řízení Báze Dat** (SŘBD) v anglickém originále **Relational Database Management System** (RDBMS).

# Historie databází

Vývoj databází velkou měrou ovlivnila vláda Spojených států amerických (její požadavky na rozsáhlé zpracování dat) a společnost IBM. Právě tato společnost jako první před více než padesáti lety vyvinula pevný disk(harddisk),který je dodnes nejpoužívanějším fyzickým médiem pro uchovávání dat v počítači.

A právě její zaměstnanec *Edgar F .Codd* v roce 1970 ve svém článku nastínil a definoval vlastnosti relačních databází, které dodnes patří k nejpoblárnějším. IBM společně s Oracle také patřila k prvním společnostem nabízejícím komerční relační SŘBD. V neposlední řadě to byla právě **IBM, která vyvinula jazyk SQL**, který se stal de facto standardem pro práci s databází.

Ač relační databáze představovaly velký krok kupředu, vývoj se nezastavil. S nástupem objektově orientovaného programování se v osmdesátých letech minulého století začaly objevovat objektové a objektově orientované systémy řízení báze dat.

Dalším typem databází, které se v současné době stále více a více prosazují, jsou takzvané **NoSQL** databáze. I přes tento pokrok jsou relační databáze stále nejpoblárnější.

# SQL (Structured Query Language)

## Jazyk pro komunikaci s databází.

Vývoj jazyka SQL (původně se nazýval SEQUEL) začal v **70. letech minulého století** (společně s rozvojem relačních databází) a během svého vývoje se stal nutnou znalostí kohokoliv, kdo, byť i jen okrajově, s databází pracoval.

Během svého vývoje se postupně standardizoval (standard **SQL-86**, **SQL-92**, **SQL-99**). Standardy tedy existují a podporuje je prakticky každá relační databáze. Bohužel ne každá implementuje všechny. Téměř každá si přidává vlastní konstrukce, které nejsou součástí standardu.

SQL se pro jednotlivé databáze se lehce liší. Hovoříme o **různých dialektech jazyka SQL**.

# Relační databáze

Databáze založené na tabulkách, kde řádky představují záznamy a sloupce vlastnosti záznamů.

- **MySQL** : Dříve nejpopulárnější databáze pro webové aplikace, široce podporovaná hostinky. Vznikla v roce 1995, nyní ve vlastnictví Oracle.
- **MariaDB** : Fork MySQL vyvíjený komunitou. Zaměřuje se na svobodný software (GNU GPL) a je binárně kompatibilní s MySQL. Dnes převažuje nad MySQL.
- **PostgreSQL** : Open Source databáze s kořeny v projektu Ingres (70. léta). První verze byla vydána v roce 1997. Pro podnikové i webové aplikace.
- **SQLite** : Embedded databáze, uložená v jednom souboru (.db). Nenáročná, ideální pro lokální data (Android, Chrome). První vydání: 2000.
- **MS SQL**: Podniková databáze od Microsoftu (od roku 1989). Nelze provozovat na Linuxu, vhodná pro Windows aplikace.

# Příklad: Vytvoření nové databáze na webhostingu

Databáze [vytvořit novou]				
název	typ	stav	velikost *	
d205431_booking	MariaDB	aktivní	3 MB	
d205431_cryo	MariaDB	aktivní	1 MB	
d205431_hdztnp	MariaDB	aktivní	0 MB	
d205431_php	MariaDB	aktivní	0 MB	
d205431_pweh5e	MariaDB	aktivní	23 MB	
d205431_w5zafu	MariaDB	aktivní	9 MB	

## Webhosting - nová databáze

Tento formulář slouží ke zřízení nové databáze k vašemu webhostingu. Databáze je zdarma, pouze je omezen celkový objem dat ve všech databázích, které patří k jednomu webhostingu. Ke zřízení dojde během několika minut po odeslání tohoto formuláře, poté obdržíte informace o zřízené databázi e-mailem.

### Nová databáze

Ke službě

edumach.cz (3218108068)

Název:

d205431\_

*libovolné pojmenování, pouze znaky a-z,0-9, max. 7 znaků*

Typ:

MariaDB (10.4.34-MariaDB-log) ▾

Příjemce e-mailu s heslem:

**Ze služby edumach.cz:**

jirimachac@gmail.com

Jiný: *více e-mailů oddělených čárkou*

vytvořit

# Jazyk SQL

SQL (*Structured Query Language*) je standardizovaný jazyk pro práci s relačními databázemi.

K čemu slouží?

- **Dotazování:** Vyhledávání dat v databázi pomocí příkazu **SELECT**.
- **Manipulaci:** Vkládání (**INSERT**), aktualizace (**UPDATE**), mazání (**DELETE**) dat.
- **Definici:** Vytváření a změna struktury databází a tabulek (**CREATE**, **ALTER**).
- **Řízení přístupu:** Nastavení uživatelských oprávnění (**GRANT**, **REVOKE**).

# Instalace MariaDB

```
$ sudo apt update
```

```
$ sudo apt install mariadb-server php-mysql
```

Vysvětlení:

- **mariadb-server** – samotný server MariaDB
- **mariadb-client** – nástroje pro příkazovou řádku (např. mysql)
- **php-mysql** – propojení PHP ↔ MariaDB/MySQL
- **php-cli, php-common, php-curl, php-xml, php-mbstring** – běžná PHP rozšíření

# Kontrola a řízení běhu

## Kontrola běhu

MariaDB se spustí jako služba (port **3306/tcp**) ihned po instalaci (jako Apache). Můžeme si to ověřit příkazem:

```
$ systemctl status mariadb
```

## Řízení běhu

Pro řízení běhu MariaDB fungují stejné řídicí direktivy jako v případě Apache:

```
$ sudo systemctl start|stop|restart|reload mariadb
```

```
root@webmr:~# systemctl status mysql
● mariadb.service - MariaDB 10.1.37 database server
   Loaded: loaded (/lib/systemd/system/mariadb.service; enabled; vendor preset:
   Active: active (running) since Mon 2018-12-03 15:03:37 CET; 18min ago
     Docs: man:mysqld(8)
           https://mariadb.com/kb/en/library/systemd/
   Process: 565 ExecStartPost=/bin/sh -c systemctl unset-environment _WSREP_START
   Process: 563 ExecStartPost=/etc/mysql/debian-start (code=exited, status=0/SUCC
   Process: 382 ExecStartPre=/bin/sh -c [ ! -e /usr/bin/galera_recovery ] && VAR=
   Process: 374 ExecStartPre=/bin/sh -c systemctl unset-environment _WSREP_START_
   Process: 354 ExecStartPre=/usr/bin/install -m 755 -o mysql -g root -d /var/run
   Main PID: 471 (mysqld)
    Status: "Taking your SQL requests now..."
     Tasks: 27 (limit: 4915)
   CGroup: /system.slice/mariadb.service
           └─471 /usr/sbin/mysqld

pro 03 15:03:32 webmr systemd[1]: Starting MariaDB 10.1.37 database server...
pro 03 15:03:35 webmr mysqld[471]: 2018-12-03 15:03:35 140155052655040 [Note] /u
pro 03 15:03:37 webmr systemd[1]: Started MariaDB 10.1.37 database server.
lines 1-19/19 (END)
```

## Systémový a databázový root

Účet správce Linuxu se jmenuje **root**. To už (doufám) víme.

Databázový server má také svého **roota** – uživatele s neomezenými právy.

**Jmenují se sice stejně, ale každý je jiný.**

V Debianu je umožněn přístup "linuxového" roota do DB systému bez hesla.

Ostatní uživatelé se do DB konzole dostanou, pokud tam mají založený uživatelský účet s dalšími nastaveními – oprávněními.

# Databázový terminál

Přihlas se do konzole MariaDB:

```
$ sudo mariadb
```

Konzole MariaDB se vám "ohlásí" **svým promptem**:

```
MariaDB [(none)]>
```

Podobně jako v linuxovém shellu lze šipkami procházet historie příkazů a v některých situacích doplňování tabulátorem.

Pro ukončení zadej příkaz **quit** nebo zkratku **Ctrl+D**.

# Terminologie

Úvodem je nutné si **vymezit některé základní pojmy** jako **dotaz**, **příkaz**, **klauzule**.

Stručně řečeno, **příkazy** jsou klíčová slova (**USE**, **SELECT**, **FROM**, **CREATE**), **dotaz** je kompletní zápis ukončený znakem **;** (středník).

Například celý zápis dotazu **USE** **nazev\_db** **;** je **dotaz**, klíčové slovo **USE** je **klauzule** (**příkaz**).

# Referenční integrita

Databáze jsou postavené na **třech hlavních pilířích**:

1. Rychlost
2. Bezpečnost
3. Referenční integrita

Referenční integrita zajišťuje, aby mezi propojenými údaji v databázi **nevznikaly nelogické nebo chybné vztahy**. Pokud jeden záznam odkazuje na jiný, musí tento odkazovaný záznam existovat.

Modelový příklad:

- V databázi jsou dvě informace – **zaměstnanec** a jeho **plat**.
- Plat je navázán na konkrétního zaměstnance. Pokud by byl zaměstnanec smazán, ale jeho plat zůstal v databázi, vznikl by nesmyslný záznam – plat bez vlastníka.
- Referenční integrita tomu zabrání – databáze buď smazání zaměstnance nedovolí, nebo automaticky odstraní i jeho plat podle nastaveného pravidla.

# Výpis databází

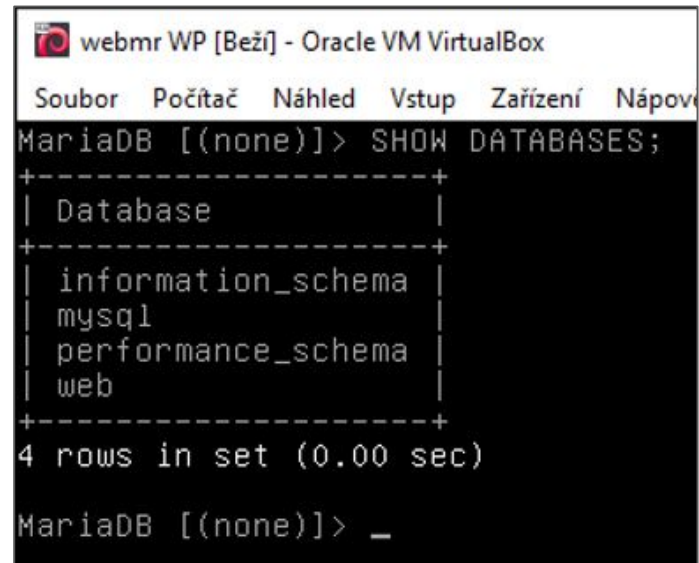
Jaké databáze máme k dispozici zjistíme SQL dotazem:

```
> SHOW DATABASES;
```

Databázový server obsahuje i systémové databáze:

```
information_schema  
mysql  
performance_schema
```

S těmi nemanipulujeme. DB server si do nich ukládá technické údaje, uživatele, oprávnění atp.



```
webmr WP [Beží] - Oracle VM VirtualBox  
Soubor Počítač Náhled Vstup Zařízení Nápov  
MariaDB [(none)]> SHOW DATABASES;  
+-----+  
| Database |  
+-----+  
| information_schema |  
| mysql |  
| performance_schema |  
| web |  
+-----+  
4 rows in set (0.00 sec)  
MariaDB [(none)]> _
```

# Založení databáze

Novou (prázdnou) databázi vytvoříme SQL dotazem:

```
> CREATE DATABASE nazev_db CHARACTER SET utf8 COLLATE utf8_czech_ci;
```

Části `CREATE DATABASE` asi rozumíme (délka názvu může být max. 65 znaků, ze speciálních znaků **pouze podtržítko** `_`), `CHARACTER SET` nastavuje znakovou sadu, `COLLATE` tzv. porovnávání. Znaková sada je soubor znaků, které může databáze obsahovat, porovnávání se většinou váže ke znakové sadě a určuje, jak se znaky za sebe řadí, tedy písmeno **ch** je mezi **h** a **i** atp. (čili `utf8_czech_ci`). Nezapomeňte na středník na konci dotazu!

Je to náš první složitější příkaz (SQL dotaz), proto několik poznámek:

- Jazyk SQL **není citlivý na velikost písmen**. Je ale zvykem:
  - Příkazy VELKÝMI PÍSMENY (lépe se odliší od zbytku dotazu).
  - Vše ostatní naopak malými písmeny.

# Nastavení aktivní databáze

Abychom mohli s nějakou databází pracovat, je vhodné (a praktické) ji nastavit jako aktivní příkazem `USE`:

```
> USE nazev_db;
```

Prompt se změní na

```
MariaDB [nazev_db]>
```

# Datové typy

Než se pustíme do vytváření tabulek, je nutné se seznámit s datovými typy. Všechny DB systémy mají tzv. **statické typování** (stejně jako jazyk C).

Datových typů je velké množství ([kompletní seznam](#)).

Pro běžné potřeby stačí tyto nejpoužívanějšími:

1. **INT** .. celá čísla ( $\pm 2$  mld. se znaménkem nebo 4 mld. bez znaménka)
2. **VARCHAR** .. text (max. 21 844 znaků ve znakové sadě UTF-8)
3. **DATE** .. datum (ve tvaru RRRR-MM-DD)

# Integritní omezení (modifikátory)

Jsou **pravidla** nastavená na úrovni databázového serveru, která zajišťují správnost a **konzistenci** dat v tabulkách. Chrání data před nesprávnými nebo neúplnými záznamy:

- **NOT NULL** – Hodnota nesmí být prázdná.
- **UNIQUE** – Hodnota musí být jedinečná.
- **DEFAULT** – Výchozí hodnota, pokud není zadána.
- **PRIMARY KEY** – Primární klíč, jehož hodnota musí být zadaná, jedinečná a jednou použitá hodnota se nesmí "recyklovat" (použít znovu). Je vždy automaticky **NOT NULL** a **UNIQUE**.
- **AUTO\_INCREMENT** – Jen pro čísla (nejčastěji **INT**). Obvykle se využívá v kombinaci s **PRIMARY KEY**.

# Uživatelé

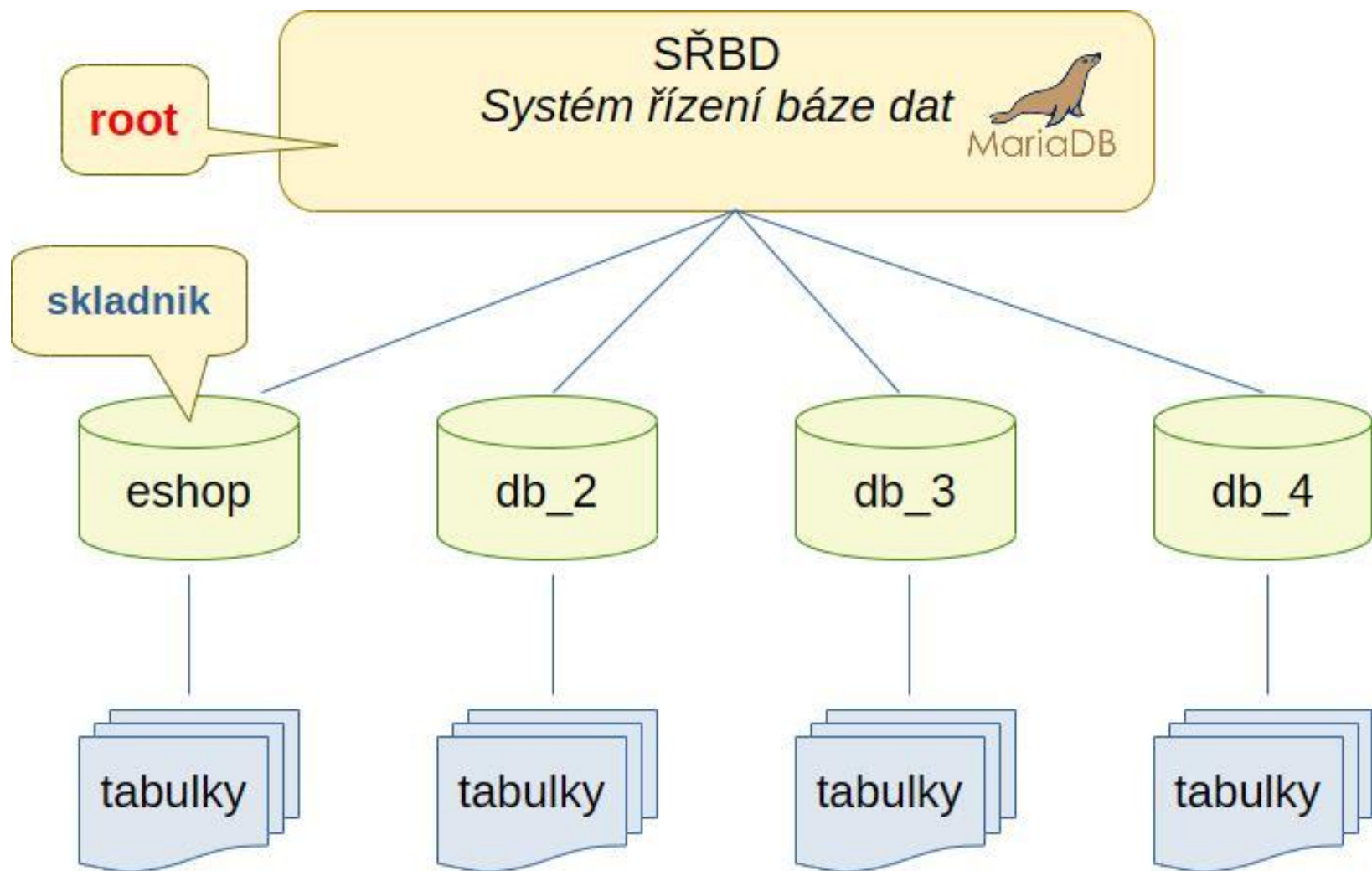
## Založení databáze a uživatele

V nově nainstalovaném MariaDB je pouze uživatel **root** (databázový root). Od Debianu verze 9 je navíc bez hesla. To má několik důsledků:

1. **Neumožňuje vzdálený přístup** (a to budeme potřebovat).
2. V rootovském terminálu se do db konzole přepnete bez hesla.

**Jako root** vytvoříme novou databázi, přidáme uživatele, přiřadíme mu heslo a patřičná oprávnění k této databázi (a pozdějším tabulkám):

- Databáze: **eshop**
- Uživatel: **skladnik**



# Vytvoření databáze a založení uživatele

```
> CREATE DATABASE eshop;
```

Založíme uživatele **skladnik** s heslem **12345**.

```
> CREATE USER 'skladnik'@'localhost' IDENTIFIED BY '12345';
```

# Přidělení oprávnění uživatel → databáze

Každý uživatel po založení nemá přidělené oprávnění k žádné databázi.

Uživateli **skladnik** oprávnění přidělíme pouze k databázi **eshop** (z důvodu bezpečnosti):

```
> GRANT ALL PRIVILEGES ON eshop.* TO 'skladnik'@'localhost';
```

Aktualizujeme uživatele a databáze v databázovém stroji MariaDB:

```
> FLUSH PRIVILEGES;
```

**Pozn:** oprávnění jde nastavit **velmi detailně**. Pro naše potřeby postačí přidělení všech (GRAND ALL) pro danou databázi.

## "Výstupní" kontrola

```
> SHOW DATABASES;
```

```
> SHOW GRANTS FOR 'skladnik'@'localhost';
```

# Tabulka

# Přihlášení jako uživatel **skladnik**

Odhlas se db konzole a přihlas už jako uživatel **skladnik**. I zde platí, že běžně pracovat pod rootem se nevyplácí.

```
$ mariadb -u skladnik -p12345 eshop  
$ mariadb -u skladnik -p eshop
```

Kdokoliv jiný kromě roota se **musí přihlásit jménem a heslem**.

1. První možnost je včetně hesla. Zadává se **bez mezery** (i mezera je znak). Toto se používá hlavně ve skriptech.
2. Druhá možnost je bez hesla. To si systém vyžádá.
3. Volitelně se může zapsat i název databáze. Odpadá tak nutnost zadávat příkaz **USE eshop;**. Zadává se bez přepínače.

## Tabulka "sklad" v databázi "eshop"

ID	Kategorie	Název	Cena	Počet
	Tričko	Tričko modré	200	6
	Tričko	Tričko zelené	250	10
	Deštník	Deštník velký	300	5
	Deštník	Deštník malý	150	2
	Kalhoty	Kalhoty dlouhé	600	7
	Kalhoty	Kalhoty krátké	300	3
	Sukně	Sukně barevná dlouhá	500	9
	Mikina	Mikina s kapucí	800	6

# Vytvoření tabulky

Dotazem pro vytvoření databázové tabulky je `CREATE TABLE`. Jeho kostra a syntaxe je následující:

```
CREATE TABLE tabulka (  
    pole1 typ [modifikatory],  
    pole2 typ [modifikatory],  
    ... ,  
    poleN typ [modifikatory]  
);
```

- Za posledním polem se samozřejmě již čárka nepíše.
- Výraz v hranatých závorkách znamená, že je nepovinný. Na některé sloupce nemusíme chtít aplikovat žádná integritní omezení.

# Ukázková databáze a tabulka

```
> USE eshop;
```

```
> CREATE TABLE sklad (  
  id int PRIMARY KEY AUTO_INCREMENT,  
  kategorie VARCHAR(30),  
  nazev varchar(100) NOT NULL UNIQUE,  
  cena int DEFAULT 0,  
  pocet int unsigned NOT NULL DEFAULT 0  
);
```

# Zobrazení struktury tabulky

> DESCRIBE sklad;

Zobrazí přehledovou tabulku názvů polí, datových typů a dalších integritních omezení.

Je důležité vědět, že struktura tabulky a vkládání dat probíhá ve všech DB systémech **odděleně**.

# Přidání záznamů do tabulky

```
INSERT INTO sklad (kategorie, nazev, cena, pocet) VALUES  
("Tričko", "Tričko modré", 200, 6);
```

```
INSERT INTO sklad (kategorie, nazev, cena, pocet) VALUES  
("Tričko", "Tričko zelené", 250, 10);
```

- Pořadí názvů polí a dat musí souhlasit.
- Text se vkládá do uvozovek, čísla nikoliv (dle datových typů polí).
- Samotná data diakritiku samozřejmě mít mohou (*Tričko modré*).
- Hodnota **ID** se nezadává, doplní se automaticky (**AUTO\_INCREMENT**).

# Jednodušší způsob s jedním **INSERT**

```
INSERT INTO sklad (kategorie, nazev, cena, pocet) VALUES
```

```
("Deštník", "Deštník velký", 300, 5),  
("Deštník", "Deštník malý", 150, 2),  
("Kalhoty", "Kalhoty dlouhé", 600, 7),  
("Kalhoty", "Kalhoty krátké", 300, 3),  
("Sukně", "Sukně barevná dlouhá", 500, 9),  
("Mikina", "Mikina s kapucí", 800, 6);
```

# Odstranění tabulky

Pokud by se něco nepovedlo, lze tabulku odstranit

**BEZ MOŽNOSTI JI OBNOVIT!**

```
> DROP TABLE sklad;
```

# Import tabulky **sklad**

Opisovat dlouhé bloky kódu, navíc v terminálu není úplně komfortní.

1. Stáhni soubor s SQL kódem,
2. prohlédni si obsah souboru (je komentovaný),
3. naimportuj ji.

```
$ wget https://github.com/edumach/eshop/blob/master/eshop.sql  
$ less eshop.sql  
$ mariadb -u skladnik -p < eshop.sql
```

# Výběrové dotazy

# Výpis záznamů

Výpis záznamů je **vůbec nejpoužívanějším dotazem**. Slouží k němu příkaz **SELECT**:

**Výpis všech záznamů:**

```
> SELECT * FROM sklad;
```

\* znamená "všechna pole v tom pořadí, jak jsou definovaná v tabulce".

**Výpis jen určitých polí:**

```
> SELECT nazev, pocet FROM sklad;
```

# Eliminace duplicitních hodnot

Některá pole často obsahují mnoho duplicitních hodnot. A někdy chcete vypisovat pouze různé (odlišné) hodnoty. Příkaz `SELECT DISTINCT` slouží k výpisu pouze odlišných (různých) hodnot. Porovnejte výstupy:

Bez DISTINCT:

```
> SELECT kategorie FROM sklad;
```

S DISTINCT:

```
> SELECT DISTINCT kategorie FROM sklad;
```

# Omezení výpisu podmínkou

V dotazech velmi často používáme podmínky pro určení, které záznamy chceme vypsat. K tomu slouží příkaz **WHERE pole operátor hodnota**.

Výpis všech produktů, kterých máme na skladě víc než pět kusů:

```
> SELECT * FROM sklad WHERE pocet > 5;
```

Výpis všech deštníků:

```
> SELECT * FROM sklad WHERE kategorie LIKE "kalhoty";
```

→ **Číselné typy:** Relační operátory jsou `<` `>` `<=` `>=` `<>` (popř. `!=`)

→ **Text:** **LIKE** znamená "obsahuje", **NOT LIKE** znamená "neobsahuje"

→ **Více podmínek** se spojují operátory **AND**, **OR** nebo **NOT**.

# Výčet hodnot

Operátor **IN** umožňuje zadat více hodnot v podmínce. Nahrazuje vícenásobné použití operátoru **OR**. Tyto dva dotazy vrátí stejný výsledek:

```
> SELECT * FROM sklad WHERE kategorie LIKE "Kalhoty" OR  
kategorie LIKE "Tričko";
```

```
> SELECT * FROM sklad WHERE kategorie IN ("Kalhoty",  
"Tričko");
```

# Řazení výpisu

K řazení slouží příkaz `ORDER BY`. Bez ničeho (nebo s `ASC`) řadí vzestupně, s `DESC` řadí sestupně:

```
> SELECT * FROM sklad ORDER BY navez;
```

Příkaz `ORDER BY` musí být až na konci:

```
> SELECT * FROM sklad WHERE cena >= 100 ORDER BY pocet DESC;
```

# Omezení počtu záznamů ve výpisu

Příkaz SELECT vždy vypisuje všechny dostupné záznamy. Jejich počet jde omezit příkazem **LIMIT**:

```
> SELECT * FROM sklad LIMIT 2;
```

Vypíše první dva záznamy.

```
> SELECT * FROM sklad LIMIT 4,2;
```

Přeskočí první 4 záznamy a následující 2 vypíše. Čili vypíše záznamy 5 a 6.

# Výpočty s daty

Názvy polí lze chápat jako proměnné. Jde s nimi provádět výpočty. Vypište si celkovou cenu položek na skladě:

```
> SELECT nazev, cena, pocet, cena * pocet FROM sklad;
```

Poslední sloupec je vypočítaný a tudíž nemá svůj název. Lze jej přidat příkazem **AS** (alias):

```
> SELECT nazev, cena, pocet, cena * pocet AS "Celkem" FROM sklad;
```

# Seskupování záznamů

Další častou potřebou je vytváření různých souhrnů. K tomu je příkaz `GROUP BY`. Pak jde zobrazit např. celkový počet kusů zboží podle kategorií:

```
> SELECT kategorie, SUM(pocet) AS "Celkem" FROM sklad GROUP BY kategorie;
```

Kromě matematických operací má MariaDB (i jiné SQL systémy) spoustu zabudovaných funkcí (jako např. Excel).

Běžné jsou `SUM()`, `COUNT()`, `MAX()`, `MIN()`, `AVG()`.

# Shrnutí výběrového dotazu

Kromě příkazů **SELECT** a **FROM** jsou ostatní nepovinné. Pokud byste použili všechny příkazy najednou, **musí být v tomto pořadí**:

**SELECT** ... (AS)

**FROM** ...

**WHERE** ...

**GROUP BY** ...

**ORDER BY** ...

**LIMIT** ...

# Úprava záznamů

# Aktualizace dat

Pro změnu dat v záznamech slouží příkaz **UPDATE**. K tomu budeme navíc potřebovat nějaký **identifikátor záznamu**, který zadáme v podmínce **WHERE**. Mohou nastat 4 stavy:

1. Podmínka vyhovuje pouze jednomu záznamu.
2. Podmínka vyhovuje více záznamům.
3. Podmínka vyhovuje všem záznamům (např. chybí **WHERE**).
4. Podmínka nevyhovuje ani jednomu záznamu.

**Dejte pozor, provedené změny nelze vzít zpět!**

Dotaz u záznamu s **ID 2** nastaví **pocet** na **20**:

```
> UPDATE sklad SET pocet = 20 WHERE id = 2;
```

Nebo můžeme snížit cenu všech **kalhot** o 10 %:

```
> UPDATE sklad SET pocet = pocet * (1 - 10 / 100) WHERE kategorie LIKE "Kalhoty";
```

# Adminer

# Adminer

Adminer je lehký, jednoduchý a velmi populární webový nástroj pro správu databází od českého vývojáře *Jakuba Vrány*. Hlavní vlastnosti:

Domovská stránka: <https://www.adminer.org>

Zdrojové kódy jsou – jak jinak – na GitHubu <https://github.com/vrana/adminer/>

Podporuje různé databázové systémy, jako MySQL, PostgreSQL, SQLite, MariaDB, MS SQL apod.

Aplikaci tvoří jediný PHP soubor, což výrazně zjednodušuje zprovoznění.

Umožňuje snadno provádět operace, jako je:

- Správa databází, tabulek, záznamů.
- Spouštění SQL dotazů.
- Export a import dat (SQL, CSV).

# Stažení Admineru

```
$ mkdir -p /var/www/html/adminer
```

```
$ wget "https://www.adminer.org/latest-mysql-cs.php" -O  
/var/www/html/adminer/index.php
```

# Přístup **roota** do Admineru

Nastavte heslo a přepněte autentizaci na heslo. V konzoli MariaDB zadejte:

```
> ALTER USER 'root'@'localhost'  
> IDENTIFIED VIA mysql_native_password  
> USING PASSWORD('SilneHeslo');  
> FLUSH PRIVILEGES;
```

Zkus přihlášení už s heslem:

```
$ mysql -u root -p
```

Pak už by mělo jít přihlásit se do Admineru **databázovým rootem**.