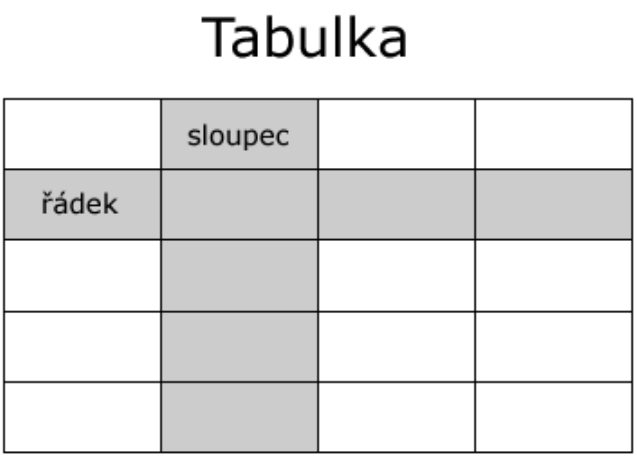
Databázový server MySQL

Databázové tabulky



Většina dnešních SŘBD je založena na principu **relačního modelu**. Tento model vychází z ukládání dat do **tabulek.** Tabulku si můžeme představit jako jakoukoliv tabulku, která obsahuje sloupce a řádky.

Sloupce - určují typ dat, který do něj (do jednotlivých řádků) můžeme vložit. Sloupce jsou nazývány **atributy**.

Řádky - jsou to jednotlivé záznamy v tabulce, které bývají identifikovány pomocí **klíčů.**

Jako příklad si můžeme ukázat tabulku, která bude obsahovat seznam učitelů.

ID	Jméno	Příjmení	Rodné číslo
1	Josef	Marný	731210/2509
2	Pavel	Novotný	710422/3583
3	Veronika	Novotná	710422/3553

Tuto tabulku můžeme využít jako vzor a vysvětlit si její strukturu:

- **ID** - Unikátní id (číslo) identifikující záznam.
- **Jméno** - Jméno osoby.
- **Příjmení** - Příjmení osoby.
- **Rodné číslo** - Rodné číslo konkrétní osoby.

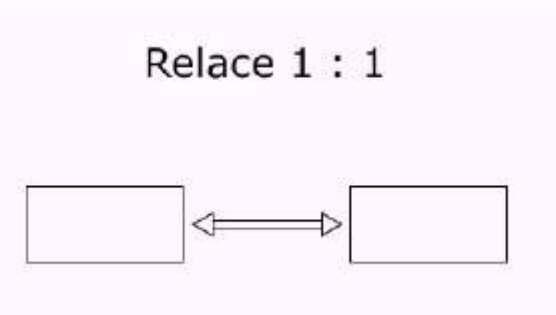
Abychom mohli jednoznačně identifikovat každý řádek, užívají se tzv. **primární klíče**. Primárním klíčem se může stát každý sloupec, který nebude mít v žádném řádku duplicitní hodnotu. Primárním klíčem se nesmí stát pole, které by mohlo obsahovat prázdnou hodnotu (NULL). V našem příkladě bychom mohli jako primární klíč zvolit sloupec **id**. Pokud se pořádně podíváme na tabulku tak zjistíme, že jako primární klíč můžeme zvolit také rodné číslo (je pro každého občana unikátní).

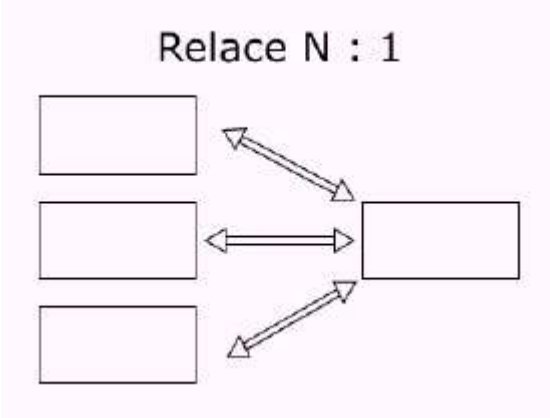
Relace

Mezi jednotlivými tabulkami mohou existovat tzv. **relace**, které mohou vytvořit vztahy mezi sloupci (poli). Můžeme tak propojit naši tabulku učitelů s tabulkou tříd, která obsahuje seznam tříd.

Relace 1 : 1

Patří mezi nejzákladnější z typů relací a umožňuje vytvořit relaci mezi dvěma tabulkami. V praxi to znamená, že právě jeden učitel bude třídním učitelem právě jedné třídy (řádek z jedné tabulky bude svázán s jedním řádkem z druhé tabulky).



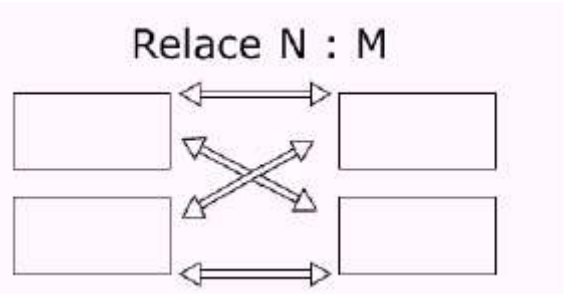


Relace N : 1

Jedná se o relaci, kdy jeden řádek z jedné (primární) tabulky bude svázán s více řádky z tabulky sekundární. Příkladem může být například to, že jeden učitel může být třídním učitelem více tříd (pozn. na školách s dálkovým studiem ta možnost existuje), ale každá třída má právě jednoho třídního učitele.

Relace N : M

Jedná se o nejkomplikovanější relaci, kdy několik řádků primární tabulky je svázáno s několika řádky sekundární tabulky. Jako příklad si můžeme uvést to, že jeden autor může napsat několik článků a jeden článek může být napsán více autory.



MySQL

MySQL patří mezi nejpopulárnější volně šiřitelné databázové servery. MySQL je populární nejen kvůli tomu, že je dostupný na mnoha platformách (Windows, Linux, FreeBSD, atd.), ale také proto, že je zdarma.

Správa MySQL

Pro správu databází existuje mnoho nástrojů. Již v MySQL distribuci nalezneme klienta, který nám umožní zpracovávat dotazy. Jedná se o konzolovou **aplikaci mysql.exe,** kterou nalezneme v adresáři bin.

Případně můžeme využít nástroj **phpMyAdmin,** který patří mezi nejznámější webové nástroje pro správu MySQL databázových serverů.

Vytváření databází

Po připojení k MySQL serveru můžeme vypsat všechny dostupné databáze:

```
SHOW DATABASES;
```

```
+----------+
| Database |
+----------+
| mysql    |
| test     |
| tmp      |
+----------+
```

Poznámka: příkazy, které si budeme uvádět můžeme psát také malými písmeny.

Pro vytvoření nové databáze použijeme příkaz, který má následující strukturu:

```
CREATE DATABASE <název databáze>;
```

Za název databáze dosadíme jméno, které bude mít nově vytvořená databáze.

Jednotlivé příkazy, které budeme vkládat musí být ukončeny středníkem. MySQL totiž umožňuje zadávat jednotlivé SQL příkazy pro přehlednost na více řádků.

Nyní si vytvoříme databázi "data", se kterou budeme pracovat:

```
CREATE DATABASE data;
```

V případě, že s touto databází budeme chtít pracovat, musíme ji vybrat:

```
USE data;
```

Vytváření tabulek

Pro vytváření tabulek slouží následující SQL příkaz:

```
CREATE TABLE <název tabulky>(
    <1. položka>   <typ>,
    <2. položka>   <typ>,
        ......
    <n. položka>   <typ>,)
```

Jednotlivé položky musí mít určeny svůj datový typ. Mezi hlavní datové typy patří:

Typ	Popis
TINYINT	Číslo v rozmezí -128 až 128.
SMALLINT	Číslo v rozmezí -32768 až 32767.
MEDIUMINT	Číslo v rozmezí -8388608 až 8388607.
INT	Číslo v rozmezí -2147483648 až 2147483647.
INT	Číslo v rozmezí -2147483648 až 2147483647.
BIGINT	Číslo v rozmezí -9223372036854775808 až 9223372036854775807.
FLOAT	Číslo v plovoucí řádové čárce.
NUMBER(a,b)	Desetinné číslo s **a** číslicemi před desetinnou čárkou a **b** desetinnými místy.
DATE	Datum ve formátu 'YYYY-MM-DD' (rok, měsíc,den).
DATETIME	Datum a čas ve formátu 'YYYY-MM-DD HH:MM:SS'.
TIME	Čas ve formátu 'HH:MM:SS'.
CHAR(n)	Textový řetězec o velikosti n (maximální počet znaků může být 255).
VARCHAR(n)	Textový řetězec o velikosti n (maximální počet znaků může být 255).

Když jsme si nyní ukázali datové typy, tak se můžeme pustit do tvorby ukázkové tabulky. Tabulka se bude jmenovat "`autori`" a bude obsahovat jména autorů knih. Jednotlivé položky budou:

- **id** - námi zvolené id autora - typ INT, AUTO_INCREMENT.
- **jmeno** - jméno autora - typ VARCHAR(30).
- **prijmeni** - příjmení autora - typ VARCHAR(30).

Jako primární klíč zvolit id autora. Položka, která je nastavena jako primární klíč nesmí mít prázdnou hodnotu - přiřadíme jí tedy vlastnost "NOT NULL". Vlastnost

"AUTO_INCREMENT" zajistí to, že u každého nového vloženého záznamu se id zvětší o jedničku.

```
CREATE TABLE autori (
id INT NOT NULL AUTO_INCREMENT,
jmeno VARCHAR(30),
prijmeni VARCHAR(30),
PRIMARY KEY (id));
```

Seznam tabulek v databázi můžeme zobrazit zadáním:

```
show tables;
```

```
+----------------+
| Tables_in_data |
+----------------+
| autori         |
+----------------+
```

Vkládání dat do tabulek

Nyní když máme vytvořenu tabulku "autori", můžeme do ní vkládat data. SQL příkaz, který zajišťuje vložení dat do tabulky má následující syntaxi:

```
INSERT INTO <název tabulky> VALUES ('<1. položka>', '<2. položka>', .., '<n. položka>')
```

Náš SQL příkaz bude vypadat například takto (prázdná hodnota id nám zajistí vložení automatického čísla - +1):

```
INSERT INTO autori VALUES ('', 'Petr', 'Doležal');
```

Mazání záznamů

Pro mazání záznamů slouží SQL příkaz DELETE:

```
DELETE FROM <název tabulky> WHERE <podmínka>
```

Vestavěné funkce

Mezi ty nejznámější patří:

Funkce	Popis
AVG(sloupec)	Vrátí průměr čísel ze zvoleného sloupce.
Count(sloupec)	Vrátí počet nenulových řádků.
Sum(sloupec)	Vrátí součet hodnot sloupce.
Min(sloupec)	Vrátí nejmenší hodnotu sloupce.
Max(sloupec)	Vrátí největší hodnotu sloupce.
Distinct(sloupec)	Vybere ze sloupce jedinečné hodnoty.

V textu není popis tvorby dotazů, protože těm je věnován samostatný materiál pro dotazování v jazyku SQL.

Literatura:

[1] Rympler P.: PHP od začátku, seriál na www.webguru.cz