



# Základní příkazy jazyka SQL

**DATABÁZE**

**2. ročník**

Mgr. Romana Badura

2023/2024

# Obsah

---

1. Úvod do jazyka SQL.
2. Datové typy (nejpoužívanější).
3. Sloupcová omezení.
4. Podkategorie jazyka SQL
  - DDL (Data Definition Language)
  - DML (Data Manipulation Language)
  - DCL (Data Control Language)
5. Příkazy podkategorie DDL
  - CREATE, ALTER, DROP
6. Příkazy podkategorie DML
  - INSERT, SELECT, UPDATE, DELETE
7. Příkazy podkategorie DCL
  - GRANT, REVOKE
8. Agregáčn  funkce.
9. Tvorba relace.
10. Virtuální tabulka (pohled).
11. Import a export dat.

# 1. Úvod do jazyka SQL

- **SQL** (*Structured Query Language*) je strukturovaný dotazovací jazyk pro práci s daty v relační databázi
- je neprocedurální s množinovým přístupem k datům a srozumitelný
- zahrnuje v sobě nástroje pro *tvorbu databází* (tabulek) a *manipulaci s daty* (vkládání dat, jejich aktualizaci, mazání a vyhledávání na základě požadavků)
- softwarově jsou SQL databáze založeny na modelu **klient-server**
- některé SQL systémy: **MariaDB**, **MySQL** (patří Oraclu), **Oracle**, **PostgreSQL** (pro unixové/linuxové systémy), **SQLite** (pro mobilní aplikace, bez nutnosti připojení na internet k SQL serveru)

# Důležité pojmy

## **Databázový server**

- speciální počítač (lokální nebo vzdálený), na kterém je uložena databáze a běží na něm SQL server

## **SQL server**

- aplikace, která běží na databázovém serveru a je schopna naslouchat a odpovídat na dotazy ze strany klienta

## **Klient**

- aplikace, která komunikuje s databázovým serverem pomocí SQL příkazů

## 2. Datové typy (nejpoužívanější)

Název datového typu	Popis
INT	- celá čísla
DECIMAL(n,d) FLOAT(n,d)	- desetinná čísla s parametry, kde "n" je celkový počet číslic včetně desetinných míst a "d" je počet desetinných míst
BIT, BOOL, TINYINT(1)	- logické hodnoty <b>0</b> (false) nebo <b>1</b> (true)
BLOB	- <i>Binary Large Object</i> - pro vkládání binárních dat (např. obrázky)
CHAR(n)	- textové řetězce s pevnou délkou - "n" má rozsah od 0 do 255 znaků
VARCHAR(n)	- textové řetězce s proměnlivou délkou - "n" má rozsah od 0 do 255 znaků
TEXT(n)	- textové řetězce s proměnlivou délkou - "n" má rozsah od 0 do 65 535 znaků

Název datového typu	Popis
DATE	<ul style="list-style-type: none"> <li>- datum ve formátu "<b>rok-měsíc-den</b>" respektive "<b>RRRR-MM-DD</b>„</li> <li>- rozsah od 1000-01-01 do 9999-12-31</li> </ul>
YEAR(n)	<ul style="list-style-type: none"> <li>- rok ve formátu "<b>RRRR</b>"</li> <li><b>YEAR(4)</b> = rozsah od 1901 do 2155,</li> <li><b>YEAR(2)</b> = rozsah od 1970 do 2069</li> </ul>
TIME	<ul style="list-style-type: none"> <li>- čas ve formátu "<b>HH:MM:SS.ss</b>" (hodiny, minuty, sekundy a milisekundy)</li> </ul>
DATETIME	<ul style="list-style-type: none"> <li>- kombinace typů <b>DATE</b> a <b>TIME</b> ve formátu <b>RRRR-MM-DD HH:MM:SS</b></li> </ul>
ENUM('item1','item2',...)	<ul style="list-style-type: none"> <li>- pole předem definovaných řetězců (<b>itemů</b>)</li> <li>- v buňce tabulky pak může být jeden z <b>itemů</b>, které jsou předdefinované</li> <li>- místo názvů '<b>item</b>' můžeme používat i jejich pořadí: <b>1</b> (místo '<b>item1</b>'), <b>2</b> (místo '<b>item2</b>')...</li> </ul>
SET('item1','item2',...)	<ul style="list-style-type: none"> <li>- pole předem definovaných řetězců (<b>itemů</b>)</li> <li>- v buňce tabulky pak může být i více z <b>itemů</b>, které jsou předdefinované</li> </ul>

# Rozdíl mezi datovým typem CHAR a VARCHAR

a) **CHAR** – datový typ pro práci s řetězcem, který má pevnou velikost

*Příklad:* **CHAR(6)** – zadáme-li řetězec "AHOJ", který obsahuje 4 znaky, bude velikost datového typu stále 6 znaků

A	H	O	J		
---	---	---	---	--	--

b) **VARCHAR** – datový typ pro práci s řetězcem, který má proměnlivou velikost

*Příklad:* např. **VARCHAR(6)** – zadáme-li řetězec "AHOJ", který obsahuje 4 znaky, velikost datového typu se přizpůsobí počtu zadaných znaků (řetězec bude mít velikost 4 znaky)

A	H	O	J
---	---	---	---

# 3. Sloupcová omezení

Název	Popis
PRIMARY KEY	- primární klíč
AUTO_INCREMENT	- automatické číslo přírůstkové
NOT NULL	- povinný údaj, který je nutné zadat
UNIQUE	- jedinečná (unikátní) hodnota
UNSIGNED	- hodnotami jsou kladná čísla nebo 0
ZEROFIL	- doplní číslo zepředu nulami
CHECK( <i>podmínka</i> )	- ošetření zadávání vstupních hodnot
DEFAULT <i>hodnota</i>	- nastavení výchozí hodnoty atributu
FOREIGN KEY( <i>fk</i> ) REFERENCES <i>tabulka(id_pk)</i>	- propojení dvou tabulek (cizí klíč odkazuje na primární klíč v druhé tabulce)



# 4. Podkategorie jazyka SQL

## SQL

### DDL

Data Definition Language

**CREATE  
ALTER  
DROP**

- slouží pro definování objektů v databázi
- jedná se o příkazy k vytváření, úpravě a mazání datových objektů

### DML

Data Manipulation Language

**INSERT  
SELECT  
UPDATE  
DELETE**

- slouží pro manipulaci s daty v databázi
- jedná se o příkazy ke vkládání získávání, aktualizaci a mazání dat z databáze

### DCL

Data Control Language

**GRANT  
REVOKE**

- slouží pro řízení přístupu k datům v databázi
- jedná se o příkazy pro správu uživatelských práv a rolí v databázi

# 5. Příkazy podkategorie DDL



**CREATE**  
**ALTER**  
**DROP**

# CREATE

- příkaz pro vytvoření databáze nebo databázové tabulky

**CREATE TABLE** *nazev\_tab* (sloupec1 dat\_typ1, sloupec2 dat\_typ2, ...);

*Příklad 1* - Vytvoření tabulky studenti:

```
CREATE TABLE studenti (  
    id_s INT PRIMARY KEY AUTO_INCREMENT,  
    prijmeni VARCHAR(30) NOT NULL,  
    jmeno VARCHAR(20),  
    vek INT,  
    trida varchar(5));
```

primární klíč (PK)

vlastnost PK, která zvyšuje hodnotu o 1 (tzv. přírůstková hodnota)

povinný údaj

*Poznámka:* Příkaz **CREATE DATABASE** bude probrán v předmětu OPS.

## Příklad 2 - Vytvoření tabulky studenti s nastavením omezení:

```
CREATE TABLE studenti (  
    id_s INT PRIMARY KEY AUTO_INCREMENT,  
    prijmeni VARCHAR(30) NOT NULL,  
    jmeno VARCHAR(20),  
    vek INT CHECK(vek>0 and vek<120),  
    trida varchar(5)) AUTO_INCREMENT = 1;
```

- nastavení omezení: **CHECK(vek>0 and vek<120)** pro atribut **vek** zajistí, že bude možné zadávat pouze hodnoty od 1 do 119
- nastavení přírůstkové hodnoty: **AUTO\_INCREMENT = 1** zajistí, že první záznam v tabulce bude začínat číslem **1**
- omezení je možné nastavit pro každý sloupec

# Výpis popisu tabulky

- po vytvoření databázové tabulky je vhodné ověřit její vlastnosti, nastavení primárního a cizího klíče, datové typy a další omezení použitím příkazu:

**DESC** `nazev_tab;` nebo **DESCRIBE** `nazev_tab;`

*Příklad:* **DESC** `studenti;`

	Field	Type	Null	Key	Default	Extra
1	id_s	int(11)	NO	PRI	<null>	auto_increment
2	prijmeni	varchar(30)	NO		<null>	
3	jmeno	varchar(20)	YES		<null>	
4	vek	int(11)	YES		<null>	
5	trida	varchar(5)	YES		<null>	

# ALTER

- příkaz pro změnu struktury nebo vlastností databázové tabulky podle zadané specifikace

**ALTER TABLE** *nazev\_tab* specifikace;

*Některé specifikace:*

... **ADD** *nazev\_sloupce* *datovy\_typ*;  
... **ADD COLUMN** *nazev\_sloupce* *datovy\_typ*;  
... **DROP COLUMN** *nazev\_sloupce*;  
... **RENAME** *novy\_nazev\_sloupce*;  
... **MODIFY** *puvodni\_nazev\_sloupce* *novy\_nazev\_sloupce*;  
... **CHANGE** *puvodni\_nazev\_sloupce* *novy\_nazev\_sloupce* *datovy\_typ*;  
... **ALTER** *nazev\_sloupce* **SET DEFAULT** *hodnota*;  
... **ALTER** *nazev\_sloupce* **DROP DEFAULT**;

*Příklady:*

**a)** přidání nového sloupce do tabulky



```
ALTER TABLE studenti ADD vaha FLOAT(5,2);
```

```
ALTER TABLE studenti ADD COLUMN vaha FLOAT(5,2);
```

 id_s	 prijmeni	 jmeno	 vek	 trida	 vaha
----------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------

**b)** přidání nového sloupce do tabulky a jeho umístění vedle sloupce **jmeno** (před sloupec **vek**)

```
ALTER TABLE studenti ADD COLUMN vyska_cm INT(3) AFTER jmeno;
```

 id_s	 prijmeni	 jmeno	 vyska_cm	 vek	 trida
---------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------






**c)** změna názvu sloupce v tabulce včetně uvedení datového typu

```
ALTER TABLE studenti CHANGE vaha hmotnost FLOAT(5,2);
```

 id_s	 prijmeni	 jmeno	 vek	 trida	 hmotnost
------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------






d) odstranění sloupce z tabulky

```
ALTER TABLE studenti DROP COLUMN hmotnost;
```

 id_s	 prijmeni	 jmeno	 vek	 trida
----------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------




e) u sloupce změny požadované parametry

```
ALTER TABLE studenti MODIFY COLUMN vek INT(3) AFTER trida;
```

 id_s	 prijmeni	 jmeno	 trida	 vek
----------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------

f) změna názvu databázové tabulky

```
ALTER TABLE studenti RENAME zaci;
```

 2a_dat
▼  tables 1
>  zaci



## g) nastavení výchozí hodnoty sloupce

```
ALTER TABLE studenti ALTER vek SET DEFAULT 15;
```

kontrola: `DESC studenti;`

Field	Type	Null	Key	Default	Extra
id_s	int(11)	NO	PRI	<null>	auto_increment
prijmeni	varchar(50)	NO		<null>	
jmeno	varchar(20)	YES		<null>	
vek	int(11)	YES		15	
trida	varchar(5)	YES		<null>	

## h) Odstranění nastavené výchozí hodnoty sloupce


```
ALTER TABLE studenti ALTER vek DROP DEFAULT;
```

Field	Type	Null	Key	Default	Extra
id_s	int(11)	NO	PRI	<null>	auto_increment
prijmeni	varchar(50)	NO		<null>	
jmeno	varchar(20)	YES		<null>	
vek	int(11)	YES		<null>	
trida	varchar(5)	YES		<null>	

# DROP

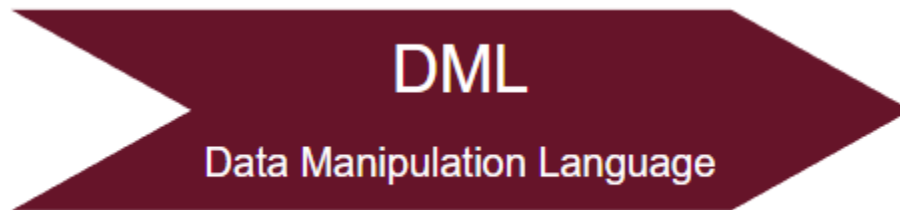
- příkaz pro odstranění databázové tabulky z databáze

**DROP TABLE** *nazev\_tab*;

*Příklad:* **DROP TABLE** studenti; >  2a\_dat



# 6. Příkazy podkategorie DML



**INSERT**  
**SELECT**  
**UPDATE**  
**DELETE**

# INSERT

- příkaz pro vkládání nových záznamů do databázové tabulky

## 1. úplné vložení (vkládáme všechny údaje)

**INSERT INTO** **nazev\_tab** **VALUES** (hodnota1, hodnota2, ...);

- pořadí hodnot je dáno pořadím sloupců v definici tabulky
- záznam obsahuje všechny údaje (je kompletní)

## 2. neúplné vložení (vkládáme některé údaje)

**INSERT INTO** **nazev\_tab** (sloupecA, sloupecB, ...) **VALUES** (hodnota1, hodnota2, ...);

- pořadí vkládání údajů je dáno seznamem sloupců za názvem tabulky v příkazu **INSERT**
- záznam neobsahuje všechny údaje (není kompletní)
- sloupce s vlastností **NOT NULL** je nutné vyplnit povinně

*Příklady:*

**a)** vložení nového záznamu do tabulky – tzv. **úplné vložení**

```
INSERT INTO studenti VALUES (DEFAULT, 'Novák', 'Jan', 16, '2.A');
```

id_s	prijmeni	jmeno	vek	trida
1	Novák	Jan	16	2.A

**b)** vložení některých údajů do tabulky (povinné, tedy s vlastností **NOT NULL** je nutné vždy zadat) – tzv. **neúplné vložení**

```
INSERT INTO studenti (prijmeni, jmeno, trida) VALUES ('Jirásek', 'Alois', '2.B');
```

id_s	prijmeni	jmeno	vek	trida
1	Novák	Jan	16	2.A
2	Jirásek	Alois	<null>	2.B

nezadaný údaj má hodnotu **NULL**

# SELECT

- příkaz pro výběr údajů z databázové tabulky

**SELECT \* FROM** **nazev\_tab;**



\* vybereme všechny sloupce z tabulky  
nebo můžeme uvést seznam sloupců, které  
pak oddělujeme čárkou

*Příklad:* **SELECT** **prijmeni**, **jmeno**, **trida** **FROM** studenti;

 id_s	 prijmeni	 jmeno	 vek	 trida
1	Novák	Jan	16	2.A
2	Jirásek	Alois	<null>	2.B
3	Dušíň	Mirek	17	2.A
4	Metelka	Jarka	17	2.B

# SELECT s podmínkou

- zobrazí se pouze ty záznamy, které splňují zadané kritérium

**SELECT \* FROM** nazev\_tab **WHERE** podmínka;

*Příklady:*

a) zobrazí se všechny sloupce z tabulky studenti a jejich hodnoty, které splňují podmínku tzn. ti, kteří navštěvují třídu 2.A

```
SELECT * FROM studenti WHERE trida like '2.A';
```

id_s	prijmeni	jmeno	vek	trida
1	Novák	Jan	17	2.A
3	Dušíň	Mírek	17	2.A

b) zobrazení některých sloupců a jejich hodnot z tabulky

```
SELECT prijmeni, jmeno FROM studenti;
```

	prijmeni	jmeno
1	Novák	Jan
2	Jirásek	Alois
3	Dušíň	Mírek



# Kritéria používaná v podmínke

Používaná kritéria	Popis
< > <= >= = !=	- porovnávací operátory
AND OR NOT &&    !	- logické operátory
IS NULL, IS NOT NULL	- porovnaní výrazů s prázdnou hodnotou
LIKE 'textwildcard' LIKE 'wildcardtextwildcard'	- % libovolné znaky, _ jeden znak
BETWEEN x AND y	- hodnoty patří do interval <x; y>
IN (seznam)	- hledá hodnoty v daném seznamu

# Speciální příkazy (klausule)

Používané klausule	Popis
<b>WHERE</b>	- určuje kritéria, podle kterých bude omezený výběr řádků z tabulky
<b>LIMIT</b>	- zobrazí požadovaný počet záznamů
<b>DISTINCT</b>	- eliminuje zobrazování duplicitních záznamů

## Příklady:

a) **LIMIT** - zobrazí první dva záznamy z databázové tabulky

```
SELECT * FROM studenti LIMIT 2;
```

id_s	prijmeni	jmeno	vek	trida
1	Novák	Jan	17	2.A
2	Jirásek	Alois	17	2.B

b) **DISTINCT** - zobrazí pouze unikátní záznamy z databázové tabulky

- Mirek Dušín se v databázové tabulce zobrazuje 2x, proto musíme zajistit, aby se zobrazoval pouze jednou

id_s	prijmeni	jmeno	vek	trida
1	Novák	Jan	17	2.A
2	Jirásek	Alois	17	2.B
3	Dušín	Mirek	17	2.A
4	Metelka	Jarka	17	2.B
5	Dušín	Mirek	17	2.A

```
SELECT DISTINCT prijmeni, jmeno, vek, trida FROM studenti;
```

prijmeni	jmeno	vek	trida
Novák	Jan	17	2.A
Jirásek	Alois	17	2.B
Dušín	Mirek	17	2.A
Metelka	Jarka	17	2.B

# Seskupování a řazení dat

Používané klauzule	Popis
<b>ORDER BY</b>	<ul style="list-style-type: none"><li>- seřadí záznamy podle sloupce(sloupců) vzestupně (<b>ASC</b>) nebo sestupně(<b>DESC</b>)</li><li>- defaultně je nastaveno vzestupné řazení <b>ASC</b></li></ul>
<b>GROUP BY</b>	<ul style="list-style-type: none"><li>- seskupí záznamy podle sloupce (sloupců)</li></ul>
<b>HAVING</b>	<ul style="list-style-type: none"><li>- filtruje záznamy na základě použití agregačních funkcí (např. <b>SUM</b>, <b>COUNT</b>, <b>AVG</b> ...)</li><li>- používá se s příkazem <b>GROUP BY</b> (záznamy musí být seskupeny)</li></ul>

## Příklady:

**a) ORDER BY** - zobrazí se záznamy seřazené podle sloupce třída (vzestupně)

```
SELECT * FROM studenti ORDER BY třída;
```

id_s	prijmeni	jmeno	vek	třída
1	Novák	Jan	17	2.A
3	Dušíň	Mírek	17	2.A
5	Dušíň	Mírek	17	2.A
2	Jirásek	Alois	17	2.B
4	Metelka	Jarka	17	2.B

**b) GROUP BY** - zobrazí pouze unikátní záznamy z databázové tabulky

```
SELECT třída, COUNT(*) FROM studenti GROUP BY třída ORDER BY třída;
```

třída	`COUNT(*)`
2.A	3
2.B	2

**c) HAVING** - jelikož žádný záznam nesplňuje kritéria podmínky, nezobrazí se nic

```
SELECT třída, COUNT(*) FROM studenti GROUP BY třída  
HAVING COUNT(vek)>17 ORDER BY třída;
```

třída	`COUNT(*)`
-------	------------

# UPDATE

- příkaz pro editaci záznamů v databázové tabulky


**UPDATE** **nazev\_tab** **SET** **sloupec** = **nova\_hodnota** **WHERE** podmínka;

- změna se (aktualizují) hodnoty pouze v těch záznamech, které splňují kritéria podmínky
- pokud neuvedeme podmínku, změna se bude týkat všech záznamů

*Příklady:*

**a)** zvýšení věku studenta o hodnotu 1 z původní hodnoty 16

```
UPDATE studenti SET vek = vek+1 WHERE prijmeni like 'Novák';
```

 id_s	prijmeni	jmeno	vek	trida
1	Novák	Jan	17	2.A

**b)** doplnění věku u studenta Aloise Jirásky

```
UPDATE studenti SET vek=17 WHERE prijmeni like 'Jirásek' AND jmeno like 'Alois';
```

 id_s	prijmeni	jmeno	vek	trida
1	Novák	Jan	17	2.A
2	Jirásek	Alois	17	2.B

# DELETE

- příkaz pro odstranění záznamů z databázové tabulky

## 1. **DELETE FROM** *nazev\_tab*;

- odstraní všechny záznamy z tabulky

## 2. **DELETE FROM** *nazev\_tab* **WHERE** podmínka;

- odstraní záznamy z tabulky, které splňují kritéria podmínky

*Příklad:*

- odstraníme z tabulky všechny studenty, kteří navštěvují třídu 2.B

```
DELETE FROM studenti WHERE trida like '2.B';
```

 id_s	 prijmeni	 jmeno	 vek	 trida
1	Novák	Jan	17	2.A
3	Dušíň	Mirek	17	2.A



# 7. Příkazy podkategorie DCL



**GRANT**  
**REVOKE**

# 8. Agregáční funkce

Název funkce	Popis
<b>AVG</b> (sloupec)	- vypočítá aritmetický průměr
<b>COUNT</b> (sloupec)	- zjistí počet hodnot ve sloupci
<b>MIN</b> (sloupec)	- určí minimální hodnotu
<b>MAX</b> (sloupec)	- určí maximální hodnotu
<b>SUM</b> (sloupec)	- vypočítá součet hodnot ve sloupci
<b>ROUND</b> (číslo)	- zaokrouhlí výsledek na celé číslo
<b>ROUND</b> (číslo,n)	- zaokrouhlí výsledné číslo na zadaný počet desetinných míst ("n")

# Další potřebné funkce

Název funkce	Popis
<b>TRIM</b> (retezec)	- odstraní mezery před a za textem
<b>LENGTH</b> (retezec)	- zjistí délku řetězce
<b>UPPER</b> (retezec)	- převede řetězec na velká písmena
<b>LOWER</b> (retezec)	- převede řetězec na malá písmena
<b>NOW</b> ()	- zjistí aktuální datum a čas
<b>CURRENT_DATE</b> ()	- zjistí aktuální datum
<b>DATE_FORMAT</b> (vstup,výstup)	- vypíše datum v určeném formátu

*Příklady:*

a) zobrazení průměrného věku studentů

```
SELECT AVG(vek) AS 'průměrný věk studentů' FROM studenti;
```

`průměrný věk studentů`
16.6667

b) zobrazení průměrného věku studentů zaokrouhleného na dvě desetinné pozice

```
SELECT ROUND(AVG(vek),2) AS 'průměrný věk studentů' FROM studenti;
```

`průměrný věk studentů`
16.67

c) zobrazení celkového počtu studentů

```
SELECT COUNT(*) AS 'celkový počet studentů' FROM studenti;
```

`celkový počet studentů`
4

# 9. Tvorba relace

## PRIMARY KEY

- nastavení primárního klíče vysvětleno na snímku 10

## FOREIGN KEY sloupec REFERENCES nazev\_tab (sloupec)

- nastavení cizího klíče vysvětleno na snímku 39)

**SELECT \* FROM tab1, tab2 WHERE** prim\_klic = cizi\_klic;

## Výpis informací z tabulek s použitím relace:

- hodnota ve sloupci primární klíč z jedné tabulky se musí shodovat s hodnotou ve sloupci cizí klíč z druhé tabulky, proto uvedeme jako kritérium podmínky rovnost obou sloupců

# Tvorba relace

## praktická ukázka

1. Vytvoříme první tabulku (nezávislou na druhé):

```
CREATE TABLE obvodni_lekar (  
    id_ol INT PRIMARY KEY AUTO_INCREMENT,  
    prijmeni_ol VARCHAR(30) NOT NULL,  
    jmeno_ol VARCHAR(30),  
    mesto_ol VARCHAR(50)  
);
```

Field	Type	Null	Key	Default	Extra
id_ol	int(11)	NO	PRI	<null>	auto_increment
prijmeni_ol	varchar(30)	NO		<null>	
jmeno_ol	varchar(30)	YES		<null>	
mesto_ol	varchar(50)	YES		<null>	

2. Vytvoříme druhou tabulku (závislou na první tabulce), kterou propojíme s první tabulkou:

```
CREATE TABLE pacient (  
    id_p INT PRIMARY KEY AUTO_INCREMENT,  
    prijmeni_p VARCHAR(30) NOT NULL,  
    jmeno_p VARCHAR(30),  
    datum_narozeni DATE,  
    lekar INT NOT NULL,  
    FOREIGN KEY (lekar) REFERENCES obvodni_lekar (id_ol)  
);
```

Field	Type	Null	Key	Default	Extra
id_p	int(11)	NO	PRI	<null>	auto_increment
prijmeni_p	varchar(30)	NO		<null>	
jmeno_p	varchar(30)	YES		<null>	
datum_narozeni	date	YES		<null>	
lekar	int(11)	NO	MUL	<null>	

# Výpis všech sloupců z obou tabulek

## 1. způsob - praktická ukázka

```
SELECT * FROM pacient, obvodni_lekar WHERE pacient.lekar=obvodni_lekar.id_ol;
```

id_p	prijmeni_p	jmeno_p	datum_narozeni	lekar	id_ol	prijmeni_ol	jmeno_ol	mesto_ol
1	Novák	Jan	1995-06-12	1	1	Ranhojič	Ivan	Havířov
2	Květáková	Marie	1972-08-15	2	2	Bolavá	Irena	Karviná
3	Výborný	František	1999-09-09	2	2	Bolavá	Irena	Karviná
4	Sojková	Miluše	1990-05-18	3	3	Slepánek	Norbert	Ostrava
5	Okurka	Antonín	1994-10-11	1	1	Ranhojič	Ivan	Havířov
6	Okurková	Alena	1985-06-30	1	1	Ranhojič	Ivan	Havířov



# Výpis všech sloupců z obou tabulek

## 2. způsob - praktická ukázka

```
SELECT id_p, pacient.prijmeni_p, pacient.jmeno_p,  
pacient.datum_narozeni, pacient.lekar, obvodni_lekar.id_ol,  
obvodni_lekar.prijmeni_ol, obvodni_lekar.jmeno_ol,  
obvodni_lekar.mesto_ol FROM pacient, obvodni_lekar  
WHERE pacient.lekar=obvodni_lekar.id_ol;
```

id_p	prijmeni_p	jmeno_p	datum_narozeni	lekar	id_ol	prijmeni_ol	jmeno_ol	mesto_ol
1	Novák	Jan	1995-06-12	1	1	Ranhojič	Ivan	Havířov
2	Květáková	Marie	1972-08-15	2	2	Bořavá	Irena	Karviná
3	Výborný	František	1999-09-09	2	2	Bořavá	Irena	Karviná
4	Sojková	Miluše	1990-05-18	3	3	Slepánek	Norbert	Ostrava
5	Okurka	Antonín	1994-10-11	1	1	Ranhojič	Ivan	Havířov
6	Okurková	Alena	1985-06-30	1	1	Ranhojič	Ivan	Havířov

# Výpis vybraných sloupců z obou tabulek

## praktická ukázka

```
SELECT id_p, pacient.prijmeni_p AS 'příjmení', pacient.jmeno_p AS 'jméno',  
pacient.datum_narozeni AS 'datum narození', obvodni_lekar.prijmeni_ol  
AS 'příjmení obvodního lékaře' FROM pacient, obvodni_lekar  
WHERE pacient.lekar=obvodni_lekar.id_ol;
```

id_p	příjmení	jméno	`datum narození`	`příjmení obvodního lékaře`
1	Novák	Jan	1995-06-12	Ranhojič
2	Květáková	Marie	1972-08-15	Bořavá
3	Výborný	František	1999-09-09	Bořavá
4	Sojková	Miluše	1990-05-18	Slepánek
5	Okurka	Antonín	1994-10-11	Ranhojič
6	Okurková	Alena	1985-06-30	Ranhojič

# 10. Virtuální tabulka (pohled)

- pohled používáme jako virtuální tabulku, která ovšem neobsahuje data
- ulehčuje nám práci se složitějším výběrem informací

## 1. Vytvoření pohledu:

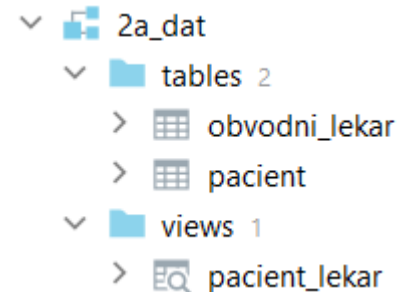
**CREATE VIEW** **nazev\_pohledu** **AS SELECT ...;**

```
CREATE VIEW pacient_lekar AS SELECT * FROM pacient, obvodni_lekar  
WHERE pacient.lekar=obvodni_lekar.id_ol;
```

## 2. Odstranění pohledu:

**DROP VIEW** **nazev\_pohledu;**

```
DROP VIEW pacient_lekar;
```



# 11. Import a export dat

- SQL umožňuje používat příkazy **pro import** nebo **export dat**
- příkazy použijeme **po přihlášení** k databázovému serveru **MariaDB**

a) příkaz pro **import dat** – SQL příkazy jsou uloženy v souboru:

**source** zaloha.txt nebo **source** zaloha.sql

b) příkazy pro **export dat** – vytvoří se záloha tabulky do souboru:

```
mysqldump --opt -u u2a01 -p db_2a01 tab1 tab2 > zaloha_tab.txt;
```

```
mysqldump --opt -u u2a01 -p db_2a01 tab1 tab2 > zaloha_tab.sql;
```

- záloha celé databáze do souboru:

```
mysqldump --opt -u u2a01 -p db_b01 > zaloha_db.txt;
```

```
mysqldump --opt -u u2a01 -p db_b01 > zaloha_db.sql;
```

# Použité zdroje

**[1]** ŠIMŮNEK, Milan. SQL: kompletní kapesní průvodce. Praha: Grada, 1999. ISBN 80-7169-692-7.

**[2]** HORDĚJČUK. [Http://voho.eu/wiki/sql/](http://voho.eu/wiki/sql/). In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-09-24]. Dostupné z: <http://voho.eu/wiki/sql/>

**[3]** [online]. [cit. 2018-10-11]. Dostupné z: <https://www.junext.net/mysql/>