

Případová studie

Návrh relační databáze

Mgr. Petr Kozák

2023

Obsah

1	Zadání a prerekvizity	3
2	Identifikace entitních typů	4
3	Identifikace vztahových typů	5
3.1	Určení kardinality a parciality vztahů pomocí diagramu výskytu vztahů.....	5
4	Určení atributů jednotlivých entitních typů.....	8
4.1	Datové typy a integritní omezení.....	8
5	E-R model	10
5.1	Lineární zápis E-R modelu před dekompozicí	10
5.2	Lineární zápis E-R modelu po dekompozici.....	10
5.3	E-R diagram	10
6	Vytvoření databáze	12
6.1	DDL SQL a terminálový monitor.....	12
6.2	SQL CREATE Script.....	13
6.2.1	SQL CREATE skript a terminálový monitor	14
6.2.2	SQL CREATE skript a phpMyAdmin	14
7	Visual Studio Code	16

1 Zadání a prerekvizity

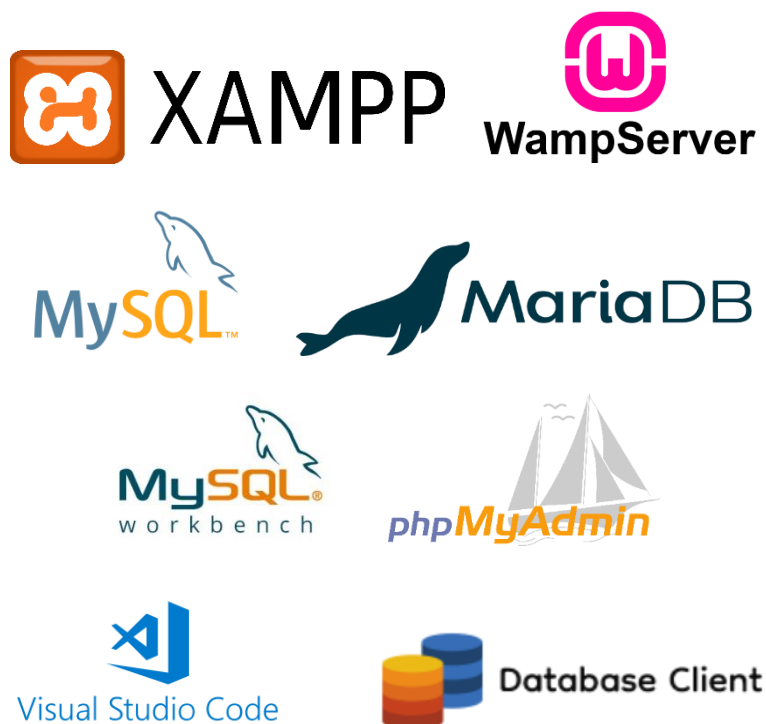
Navrhni databázi evidence vzdělávacích kurzů pro veřejnost, ve kterých školitelé školí frekventanty kurzu, a to v těchto krocích:

- identifikace entitních typů
- identifikace vztahových typů
- určení kardinality a parciality vztahů pomocí diagramu výskytu vztahů
- určení atributů jednotlivých entitních typů
- datové typy a integritní omezení
- lineární zápis E-R modelu
- E-R diagram
- vytvoření databáze

Předpokládané znalosti a dovednosti:

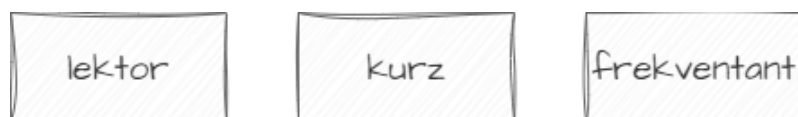
- znalost základní terminologie a principů tvorby relačních databází
- základní znalost jazyka SQL
- ovládání správy databází pomocí rozhraní, např. phpMyAdmin
- základní ovládání nástroje MySQL Workbench pro návrh a modelování relačních databází

Doporučený software:



2 Identifikace entitních typů

Z hlediska zadání potřebujeme evidovat údaje o vzdělávacích kurzech, lektorech, kteří tyto kurzy vedou, a frekventantech, kteří je absolvují. Identifikovali jsme tedy tři entitní typy: *lektor*, *kurz* a *frekventant*.



Obrázek 1 - entitní typy lektor, kurz a frekventant

Pro lepší pochopení je níže zobrazena tabulka reprezentující *záznamy* entitního typu *lektor*. Každý řádek (*záznam*, *n-tice*, *entita*) představuje informaci o jednom konkrétním lektorovi v tabulce *lektor*. Každý sloupec představuje jednu vlastnost (*atribut*) všech lektorů popsanou v záhlaví tabulky, např. *firstname*.

lektor		
id	firstname	lastname
1	Jan	Novák
2	Petr	Kolečko
3	Iva	Horáková
4	Marta	Kašková
5	Jiří	Horák

Obrázek 2 - tabulka reprezentující záznamy entitního typu lektor

Stejným způsobem jsou zobrazeny tabulky *frekventant* a *kurz*.

frekventant		
id	firstname	lastname
1	Josef	Horáček
2	Ludmila	Malá
3	Karel	Vyskočil
4	Michal	Bobek
5	Lukáš	Vytlačil

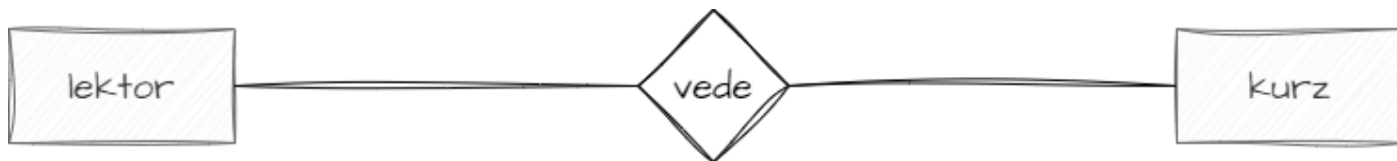
Obrázek 3 - tabulka frekventantů

kurz		
id	name	description
1	CHAT	CHAT essentials
2	AI	AI principles
3	IoT	IoT essentials
4	Wearables	W. for experts
5	VoIP	VoIP essentials

Obrázek 4 - tabulka kurzů

3 Identifikace vztahových typů

Při identifikaci entitních typů jsme mimoděk identifikovali také vztahové typy *lektori vedou kurzy* a *frekventanti absolvují kurzy*. Pro popis vztahového typu používáme většinou sloveso vyjadřující vztah mezi entitními typy.



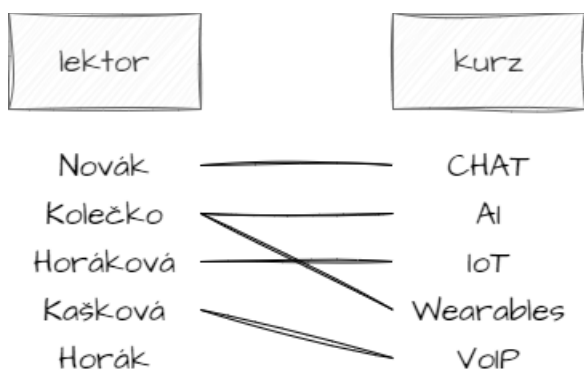
Obrázek 5 - Vztah „lektor vede kurz“



Obrázek 6 - Vztah „frekventant absolvuje kurz“

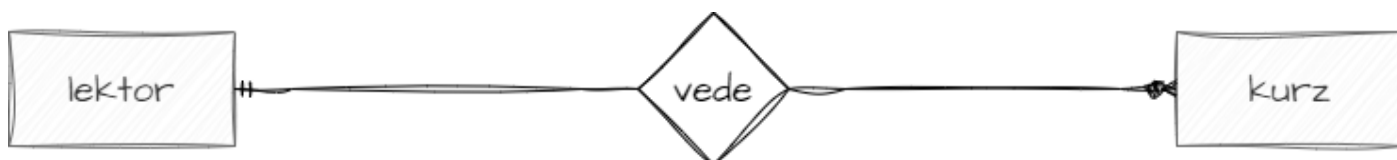
3.1 Určení kardinality a parciality vztahů pomocí diagramu výskytu vztahů

Diagram výskytu vztahů pomáhá pochopit, jaké vztahy mohou nastat z hlediska četnosti (*kardinality*) a povinnosti (*parciality*). Z diagramu níže vyplývá, že z hlediska kardinality je každý kurz veden jedním lektorem a současně každý lektor vede více kurzů. Jedná se tedy o vztah 1:N. V případě parciality nemusí lektor vést žádný kurz (nepovinný vztah), ale každý kurz musí být veden lektorem (povinný vztah).



Obrázek 7 - diagram výskytů vztahu lektor vede kurz

Znázornění vztahu z hlediska *kardinality* a *parciality* je zobrazeno níže pomocí *diagramu*. Způsoby vyjádření se mohou lišit podle použité *notace* (standardizované metody zobrazení).



Obrázek 8 - popis vztahu „lektor vede kurz“ z hlediska kardinality a parciality

Kardinalita a parcialita vztahů		
vztah	kardinalita	parcialita
$\# \text{ --- } \circ \leq$	$1 \dots N$	doleva povinný (1), doprava nepovinný (0..N)
$\geq \circ \text{ --- } \circ \leq$	$M \dots N$	doleva nepovinný (0..N), doprava nepovinný (0..N)

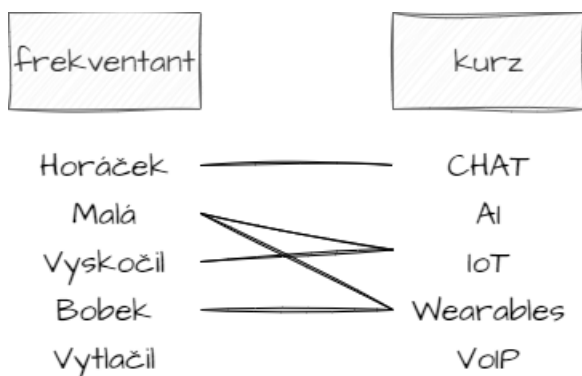
Obrázek 9 - popis grafického vyjádření vztahů z hlediska kardinality a parciality

Jak si představit konkrétní vyjádření vztahů *lektor vede kurz* pomocí tabulky? Budeme-li vycházet z diagramu výskytů vztahu *lektor vede kurz* jako modelového případu, vztahy budou zajištěny přidáním atributu *id_lektor* (cizí klíč, odkazující se na *primární klíč id* v tabulce *lektor*) tak, jak to znázorňuje tabulka níže.

kurz				lektor		
id	name	description	id_lektor	id	firstname	lastname
1	CHAT	CHAT essentials	1	1	Jan	Novák
2	AI	AI principles	2	2	Petr	Kolečko
3	IoT	IoT essentials	3	3	Iva	Horáková
4	Wearables	W. for experts	2	4	Marta	Kásková
5	VoIP	VoIP essentials	4	5	Jiří	Horák

Obrázek 10 - znázornění příkladu vztahu „lektor vede kurz“ pomocí relačních tabulek

Diagram výskytu vztahů níže ukazuje, že z hlediska kardinality každý frekventant absolvuje 0, 1 nebo více kurzů a současně každý kurz je absolvován 0, 1 nebo více frekventanty. Jedná se tedy o vztah M:N. V případě parciality nemusí frekventant absolvovat žádný kurz (nepovinný vztah) a kurz nemusí být absolvován žádným frekventantem (nepovinný vztah).



Obrázek 11 – diagram výskytu vztahů „frekventant absolvuje kurz“

Znázornění vztahu z hlediska *kardinality* a *parciality* je opět vyjádřeno níže pomocí *diagramu*.



Obrázek 12 - popis vztahu „frekventant absolvuje kurz“ z hlediska kardinality a parciality

Protože se jedná o vztah typu M:N, je nyní nutné provést dekompozici tohoto vztahu na dva vztahy 1:N.



Obrázek 13 - dekompozice vztahu „frekventant absolvuje kurz“



Obrázek 14 - dekompozice vztahu „frekventant absolvuje kurz“ s popisem relací

Pro lepší názornost vyjádříme tyto dva vztahy pomocí relačních tabulek. Vztahy budou zajištěny přidáním relační tabulky *kurzy_frekvantantu*. Tato relační tabulka obsahuje dva atributy (*id_kurz*, *id_frekvantantu*), které společně tvoří tzv. *složený primární klíč*. Tento složený primární klíč zajistí, že záznam o kurzu frekventanta bude v tabulce *kurzy_frekvantantu* nejvýše jednou.

kurz				kurzy_frekvantantu		frekventant		
id	name	description	id_lector	id_kurz	id_frekvantantu	id	firstname	lastname
1	CHAT	CHAT essentials	1	1	1	1	Josef	Horáček
2	AI	AI principles	2	3	2	2	Ludmila	Malá
3	IoT	IoT essentials	3	3	3	3	Karel	Vyskočil
4	Wearables	W. for experts	2	4	2	4	Michal	Bobek
5	VoIP	VoIP essentials	4	4	4	5	Lukáš	Vytlačil

Obrázek 15 - znázornění příkladu vztahu „frekventant absolvuje kurz“ pomocí relačních tabulek

Z grafického znázornění výše je patrné, že pomocí relační tabulky *kurzy_frekvantantu* je propojen kurz s *id* = 3 (zelené šrafování) s frekventantem, který má *id* = 2 (fialové šrafování), tedy že kurz *IoT* absolvuje *Ludmila Malá*. Červeně jsou zvýrazněny primární a cizí klíče.

4 Určení atributů jednotlivých entitních typů

Atributem entitního typu je jeho vlastnost, která je pojmenována, např. vlastností lektora je jeho příjmení – atribut *lastname*. Hodnoty této vlastnosti jsou zaznamenány v relační tabulce *lektor* ve sloupci *lastname*.



Obrázek 16 - atributy entitních typů

4.1 Datové typy a integritní omezení

Každý atribut má definován datový typ a stanovena integritní omezení tak, jak to pro každou tabulku znázorňují obrázky níže. Popis jednotlivých přepínačů integritních omezení:

- **PK** – primární klíč, jedinečný identifikátor každého záznamu
- **NN** – hodnota atributu nesmí být prázdná (pozn.: hodnota 0 není prázdná hodnota)
- **UQ** – unikátní atribut – hodnoty atributu se nesmí ve sloupci opakovat
- **B** – binární data
- **UN** – bezznaménkový datový typ (nezáporná čísla)
- **ZF** – doplnění numerického atributu nulami zleva na stejný počet cifer
- **AI** – automatická inkrementace atributu při přidávání záznamů
- **G** – generovaný sloupec - hodnoty se počítají z výrazu zahrnutého v definici sloupce

V tabulce *lektor* jsou tři atributy. Atribut *id* je datového typu *INT* (celé číslo), jedná se o primární klíč (*PK*), hodnota atributu nesmí být prázdná (*NN*), jde o bezznaménkový datový typ (*UN*) s aktivní automatickou inkrementací (*AI*). Atribut *id* je jedinečným identifikátorem záznamů v tabulce *lektor* a je využíván při vytváření vztahů mezi tabulkami.

Atribut *firstname* je textový řetězec s proměnnou délkou o maximálním počtu 45 znaků nezávisle na použitém kódování (*VARCHAR(45)*). Hodnotami atributu budou křestní jména lektorů, hodnota atributu nesmí být prázdná (*NN*). Také atribut *lastname* je textový řetězec s proměnnou délkou o maximálním počtu 45 znaků nezávisle na použitém kódování (*VARCHAR(45)*). Hodnotami atributu budou příjmení lektorů, hodnota atributu nesmí být prázdná (*NN*).








Table Name:

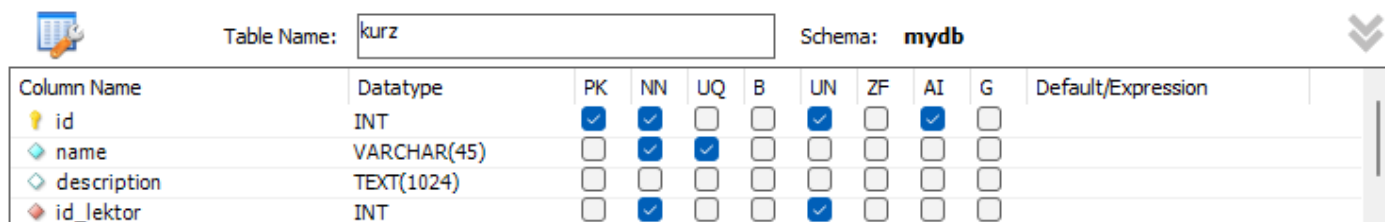
Schema: **mydb**



Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
 id	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
 firstname	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 lastname	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Obrázek 17 - atributy tabulky "lektor" (MySQL Workbench)

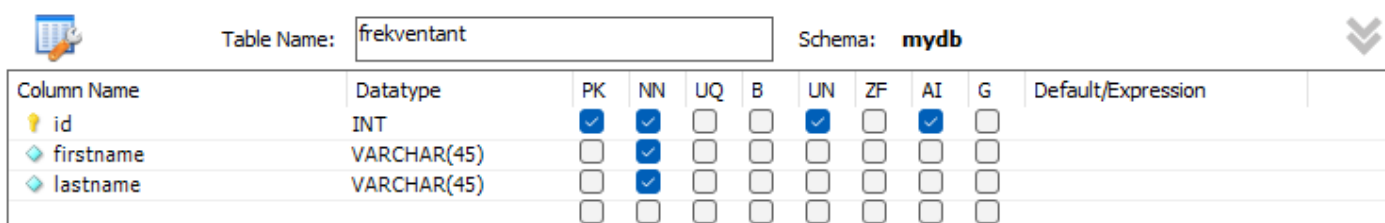
Atribut *name* tabulky *kurz* je unikátní (UQ), protože předpokládáme, že název kurzu jedinečný, tedy že různé kurzy nebudou mít stejný název. Atribut *description* je typu *TEXT* o maximálně 1024 znacích. Atribut *id_lector* je cizí klíč, který se odkazuje na primární klíč *id* z tabulky *lector*. Oba tyto klíče musí být stejného datového typu.



Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
name	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
description	TEXT(1024)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
id_lector	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Obrázek 18 - atributy tabulky "kurz" (MySQL Workbench)

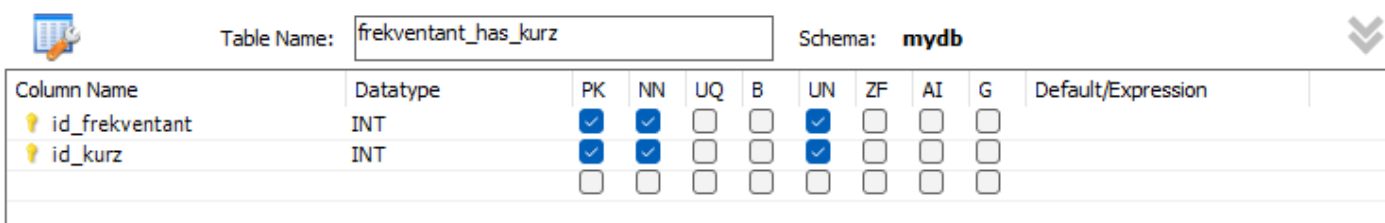
Atributy tabulky *frekventant* jsou z hlediska datových typů a integritních omezení identické s atributy v tabulce *lector*.



Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
firstname	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
lastname	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Obrázek 19 - atributy tabulky "frekventant"

frekventant_has_kurz je pomocná tabulka, která slouží k propojení záznamů v tabulkách *frekventant* a *kurz* a je prostředkem pro dekompozici vztahu *M:N* mezi oběma tabulkami na dva vztahy *1:N*. Tabulka *frekventant_has_kurz* obsahuje dva cizí klíče, z nichž *id_frekvantant* se odkazuje na primární klíč *id* z tabulky *frekventant* a *id_kurz* se odkazuje na primární klíč *id* z tabulky *kurz*. Společně atributy v tabulce *frekventant_has_kurz* tvoří tzv. složený primární klíč.



Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id_frekvantant	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
id_kurz	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Obrázek 20 - atributy tabulky "frekventant_has_kurz"

5 E-R model

Entitně-vztahový model (E-R model) se používá pro *abstraktní a konceptuální znázornění dat*. *Entity-relationship diagramy* (ER diagramy, ERD) jsou vizuálním znázorněním E-R modelu. Existuje více konvencí pro tvorbu diagramů, my v tomto konkrétním případě používáme notaci *Crow's Foot*.

V naší případové studii je cílem vytvořit *konceptuální model dat pro databázi vzdělávacích kurzů pro veřejnost*. Již jsme identifikovali entitní typy, vztahové typy, kardinalitu a parcialitu vztahů, provedli dekompozici vztahu M:N, definovali atributy, jejich datové typy a integritní omezení.

5.1 Lineární zápis E-R modelu před dekompozicí

Pro lepší názornost lze to podstatné, co jsme doposud určili, zapsat tzv. *lineárním zápisem E-R modelu*, který popisuje entitní typy a jejich atributy a vztahové typy. Z lineárního zápisu však nejsou patrné informace o kardinalitě a parcialitě vztahů, datové typy a integritní omezení.

Před provedením dekompozice vztahu M:N vypadá lineární zápis následovně:

```
E: lektor(id, firstname, lastname); frekventant(id, firstname, lastname);  
kurz(id, firstname, lastname, id_lektor).  
R: vede(lektor, kurz); absolvuje(frekventant, kurz).
```

Obrázek 21 - lineární zápis E-R modelu před dekompozicí

5.2 Lineární zápis E-R modelu po dekompozici

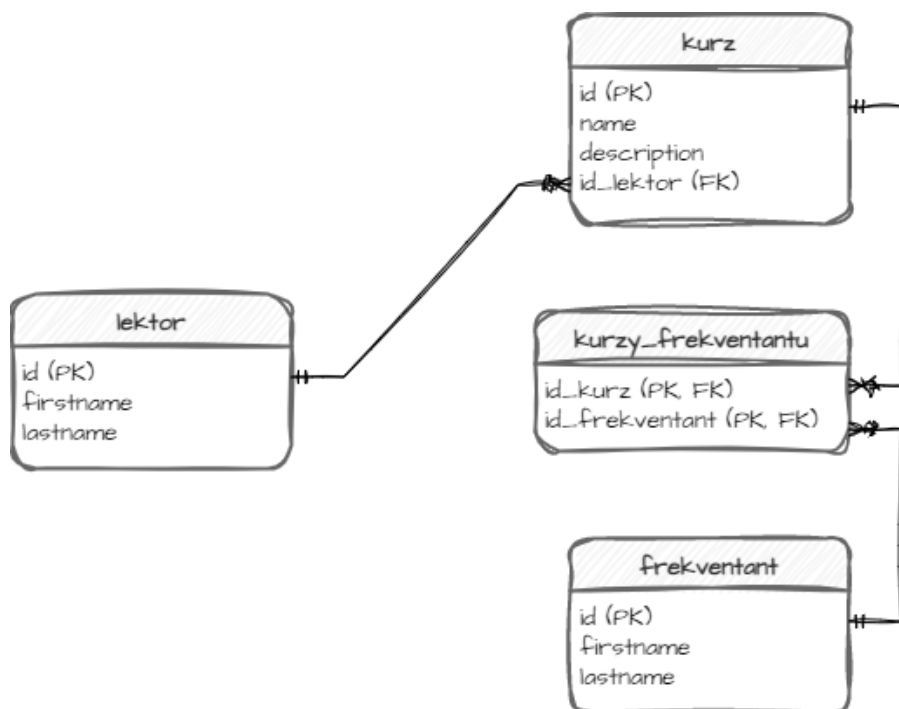
Po provedení dekompozice vztahu M:N vypadá lineární zápis takto:

```
E: lektor(id, firstname, lastname); frekventant(id, firstname, lastname);  
kurz(id, firstname, lastname, id_lektor);  
kurzy_frekventantu(id_kurz, id_frekventant).  
R: vede(lektor, kurz); absolvuje(frekventant, kurzy_frekventantu);  
je_absolvovan(kurzy_frekventantu, kurz).
```

Obrázek 22 - lineární zápis E-R modelu po dekompozici

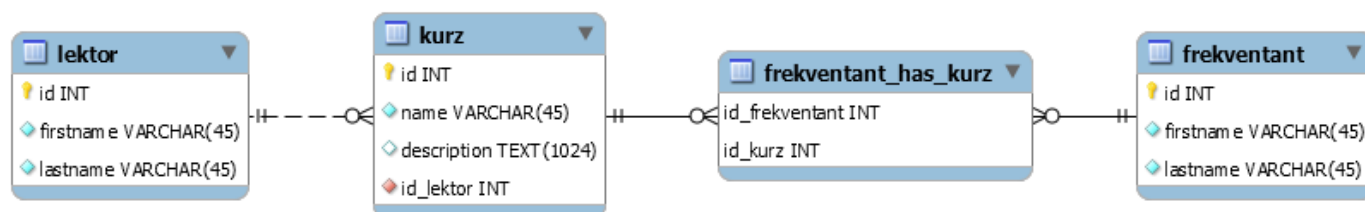
5.3 E-R diagram

Následující náčrt E-R diagramu znázorňuje entitní typy *lektor*, *kurz*, *frekventant_has_kurz* a *frekventant* a jejich atribut. Vztahové typy jsou vyjádřeny pomocí notace *Crow's Foot*. Mezi tabulkami *lektor* a *kurz* je vztah 1:N, mezi tabulkami *kurz* a *frekventant* je vztah M:N rozdělený dekompozicí na dva vztahy 1:N s využitím pomocné tabulky *kurzy_frekventantu*.



Obrázek 23 – náčrt E-R diagramu návrhu databáze

Takto vypadá *konceptuální model dat pro databázi vzdělávacích kurzů pro veřejnost* v podobě ERD navržený pomocí nástroje *MySQL Workbench*. V návrhu jsou patrné entitní typy *lektor*, *kurz*, *frekventant_has_kurz* a *frekventant* a jejich atributy včetně datových typů. Vztahové typy jsou vyjádřeny pomocí notace *Crow's Foot* stejně jako v náčrtu.



Obrázek 24 - E-R diagram návrhu databáze pomocí nástroje *MySQL Workbench*

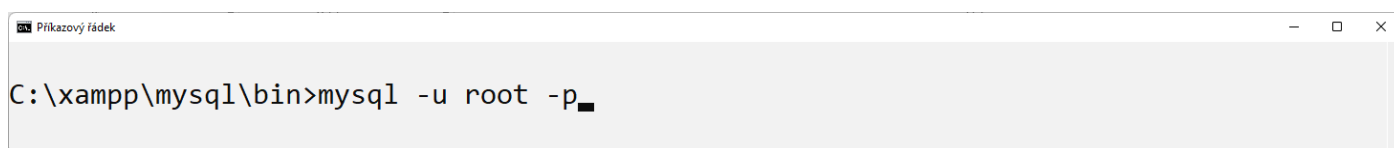
6 Vytvoření databáze

Možností, jak postupovat při vytváření databáze je mnoho, zde jsou uvedeny některé z nich:

- použití jazyka *DDL SQL* (Data Definition Language) v *MySQL/MariaDB terminálovém monitoru* – nutná znalost syntaxe jazyka *DDL SQL* a ovládání terminálového monitoru
- použití jazyka *DDL SQL* ve vybraném nástroji pro správu DB serveru, např. *phpMyAdmin* – nutná znalost syntaxe jazyka *DDL SQL* a rozhraní *phpMyAdmin*
- využití rozhraní nástroje *phpMyAdmin* a jeho funkcí pro tvorbu databází – nutná znalost rozhraní *phpMyAdmin*
- využití *forward engineeringu* v nástroji *MySQL Workbench* a exportování *SQL CREATE* skriptu s jeho následným importem přes *phpMyAdmin* nebo *terminálový monitor* – problém s kompatibilitou generovaného skriptu pro různé verze SQL serveru

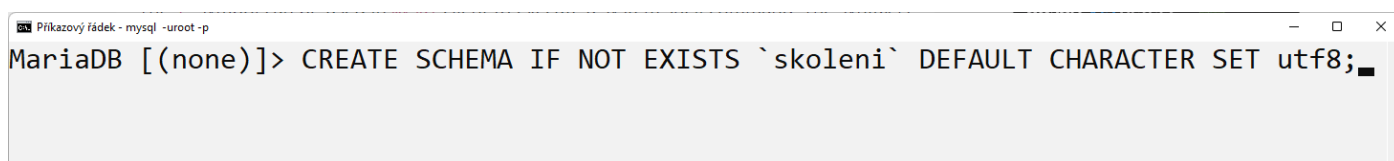
6.1 DDL SQL a terminálový monitor

Přihlášení přes terminálový monitor se provádí příkazem `mysql`. Přepínač `-u <uživatel>` určuje uživatele, pod kterým mám být přihlášení provedeno, přepínač `-p` je pro provedení požadavku na zadání hesla.



Obrázek 25 - spuštění terminálového monitoru

Pro vytvoření databáze je nutné znát syntaxi jazyka *DDL SQL*. Následující dotaz vytvoří databázi *skoleni* s defaultní znakovou sadou *utf8*. V případě, že již tato databáze existuje, dotaz nebude proveden.



Obrázek 26 - vytvoření databáze pomocí SQL dotazu v terminálovém monitoru

Chceme-li nad konkrétní databází v terminálovém monitoru provádět dotazy, je vhodné nejprve změnit aktivní databázi pomocí dotazu `USE <databáze>`. Aktivní databáze se zobrazuje v promptu monitoru.



Obrázek 27 - změna aktivní databáze pomocí terminálového monitoru

Po vytvoření celé databáze je např. možné provést zobrazení všech jejích tabulek pomocí dotazu `SHOW TABLES`.

```
Príkazový řádek - mysql -uroot -p
MariaDB [skoleni]> SHOW TABLES;
+-----+
| Tables_in_skoleni |
+-----+
| frekventant        |
| frekventant_has_kurz |
| kurz               |
| lektor             |
+-----+
4 rows in set (0.001 sec)
```

Obrázek 28 - zobrazení tabulek vybrané databáze pomocí terminálového monitoru

Zobrazení obsahu tabulky můžeme provést např. dotazem `SELECT * FROM <tabulka>`. Níže je zobrazen obsah tabulky `kurz` včetně názvů jednotlivých atributů.

```
Príkazový řádek - mysql -uroot -p
MariaDB [skoleni]> SELECT * FROM `kurz`;
+-----+-----+-----+-----+
| id | name      | description          | id_lektor |
+-----+-----+-----+-----+
| 1  | CHAT      | CHAT essentials      | 1          |
| 2  | AI        | AI principles         | 2          |
| 3  | IoT       | IoT essentials        | 3          |
| 4  | Wearables | Wearables for experts | 2          |
| 5  | VoIP      | VoIP essentials       | 4          |
+-----+-----+-----+-----+
```

Obrázek 29 - zobrazení záznamů tabulky "kurz" pomocí SQL dotazu v terminálovém monitoru

6.2 SQL CREATE Script

Vytvoření databáze lze provést importováním SQL CREATE skriptu (*File – Export – Forward Engineer SQL CREATE Script*) generovaného z *MySQL Workbench* pomocí forward engineeringu. Níže je uveden upravený zdrojový kód generovaného SQL skriptu pro server *MariaDB*.

```
-- Schema skoleni
--
CREATE SCHEMA IF NOT EXISTS `skoleni` DEFAULT CHARACTER SET utf8;
USE `skoleni`;

-- Table `skoleni`.`lektor`
--
CREATE TABLE IF NOT EXISTS `skoleni`.`lektor` (
  `id` INT UNSIGNED NOT NULL AUTO_INCREMENT,
  `firstname` VARCHAR(45) NOT NULL,
  `lastname` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;

-- Table `skoleni`.`kurz`
--
CREATE TABLE IF NOT EXISTS `skoleni`.`kurz` (
  `id` INT UNSIGNED NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
```

```

`description` TEXT(1024) NULL,
`id_lektor` INT UNSIGNED NOT NULL,
PRIMARY KEY (`id`),
INDEX `fk_kurz_lektor1_idx` (`id_lektor` ASC),
UNIQUE INDEX `name_UNIQUE` (`name` ASC),
CONSTRAINT `fk_kurz_lektor1`
  FOREIGN KEY (`id_lektor`)
  REFERENCES `skoleni`.`lektor` (`id`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION)
ENGINE = InnoDB;

-----
-- Table `skoleni`.`frekventant`
-----

CREATE TABLE IF NOT EXISTS `skoleni`.`frekventant` (
  `id` INT UNSIGNED NOT NULL AUTO_INCREMENT,
  `firstname` VARCHAR(45) NOT NULL,
  `lastname` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;

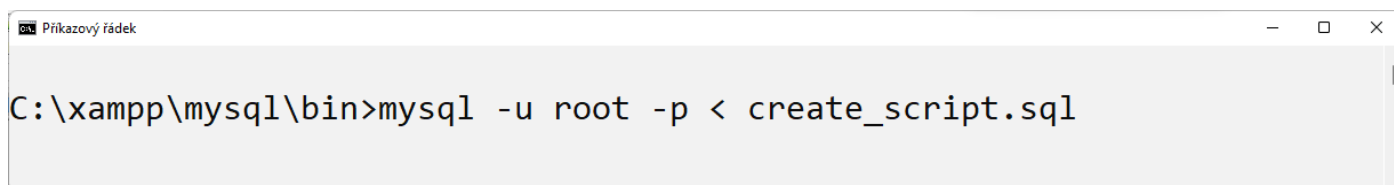
-----
-- Table `skoleni`.`frekventant_has_kurz`
-----

CREATE TABLE IF NOT EXISTS `skoleni`.`frekventant_has_kurz` (
  `id_frekvantant` INT UNSIGNED NOT NULL,
  `id_kurz` INT UNSIGNED NOT NULL,
  PRIMARY KEY (`id_kurz`, `id_frekvantant`),
  INDEX `fk_frekvantant_has_kurz_kurz1_idx` (`id_kurz` ASC),
  INDEX `fk_frekvantant_has_kurz_frekvantant_idx` (`id_frekvantant` ASC),
  CONSTRAINT `fk_frekvantant_has_kurz_frekvantant`
    FOREIGN KEY (`id_frekvantant`)
    REFERENCES `skoleni`.`frekventant` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_frekvantant_has_kurz_kurz1`
    FOREIGN KEY (`id_kurz`)
    REFERENCES `skoleni`.`kurz` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

6.2.1 SQL CREATE skript a terminálový monitor

Ukázka vytvoření databáze pomocí terminálového monitoru je zobrazena na obrázku níže. Pomocí symbolu < je po přihlášení do terminálového monitoru provedeno přesměrování *SQL skriptu* na jeho vstup.



Obrázek 30 - přesměrování SQL skriptu na vstup terminálového monitoru

6.2.2 SQL CREATE skript a phpMyAdmin

phpMyAdmin je bezplatný softwarový nástroj napsaný v jazyce *PHP*, určený ke správě *MySQL* serveru přes webové rozhraní. *phpMyAdmin* podporuje širokou škálu operací nad servery *MySQL* a *MariaDB*. Často používané operace (správa databází, tabulek, sloupců, vztahů, indexů, uživatelů, oprávnění atd.) lze provádět prostřednictvím grafického uživatelského rozhraní, přičemž je zde stále možnost přímo spouštět jakýkoli dotaz *SQL*.

Tabulka	Operace	Rádků	Typ	Porovnávání	Velikost	Navíc
<input type="checkbox"/> frekventant	★ Projít Struktura Vyhledávání Vložit Vyprázdnit Odstranit	0	InnoDB	utf8_general_ci	16,0 KiB	-
<input type="checkbox"/> frekventant_has_kurz	★ Projít Struktura Vyhledávání Vložit Vyprázdnit Odstranit	0	InnoDB	utf8_general_ci	48,0 KiB	-
<input type="checkbox"/> kurz	★ Projít Struktura Vyhledávání Vložit Vyprázdnit Odstranit	0	InnoDB	utf8_general_ci	48,0 KiB	-
<input type="checkbox"/> lektor	★ Projít Struktura Vyhledávání Vložit Vyprázdnit Odstranit	0	InnoDB	utf8_general_ci	16,0 KiB	-
4 tabulky	Celkem	0	InnoDB	utf8_general_ci	128,0 KiB	0 B

Obrázek 31 - náhled seznamu tabulek databáze "skoleni" v phpMyAdmin

Import *SQL CREATE skriptu* je možné v *phpMyAdmin* provést dvěma základními způsoby. Na záložce *SQL* lze vložit zkopírovaný text skriptu a dotazy potvrdit tlačítkem *Proved'*.

Spustit SQL dotaz(y) na serveru „127.0.0.1“:

```

1  -- -----
2  -- Schema `skoleni`
3  -- -----
4  CREATE SCHEMA IF NOT EXISTS `skoleni` DEFAULT CHARACTER SET utf8;
5  USE `skoleni`;
6
7  -- -----
8  -- Table `skoleni`.`lektor`
9  -- -----

```

Vyčistit
Formát
Načíst automaticky uložený dotaz

☐ Svázané parametry

Přidat tento SQL dotaz do oblíbených:

Oddělovač: ;
☐ Zobrazit zde tento dotaz znovu
☐ Zachovat pole pro dotaz
☐ Po dokončení vrátit zpět
☒ Zapnout kontrolu cizích klíčů
Proved'

Obrázek 32 - vytvoření databáze zkopírováním *SQL CREATE skriptu* do *SQL formuláře* v *phpMyAdmin*

Vhodnějším způsobem je import *SQL CREATE skriptu* na záložce *Import*.

Importuji na aktuální server

Soubor pro importování:

Soubor může být komprimovaný (gzip, bzip2) nebo nekomprimovaný.
Název komprimovaného souboru musí končit na **[formát].[komprese]**. Například: **.sql.zip**

Procházet váš počítač: (Maximální velikost: 40MiB)

Vybrat soubor
skoleni.sql

Můžete také přetáhnout a pustit soubor na libovolné stránce.

Znaková sada souboru:

utf-8

Obrázek 33 - Import *SQL CREATE skriptu* v *phpMyAdmin*

7 Visual Studio Code

VS Code je free software. Jedná se o výkonný editor zdrojového kódu a je k dispozici pro OS Windows, MacOS a Linux. Má bohatý ekosystém rozšíření pro další jazyky a moduly runtime, mezi které patří také podpora práce s databázovými servery MySQL. V náhledu níže je Visual Studio Code pomocí rozšíření MySQL připojen k databázovému serveru na adrese 127.0.0.1 a portu 3306. Náhled zobrazuje výběr lektorů kurzů v databázi *skoleni*.

The screenshot shows the Visual Studio Code interface with a MySQL database connection. The left sidebar displays the database structure, including tables like `frekventant`, `frekventant_has_kurz`, `kurz`, and `lektor`. The main editor displays a SQL query: `SELECT kurz.id, kurz.name, kurz.description, lektor.firstname, lektor.lastname FROM kurz JOIN lektor ON kurz.id_lektor = lektor.id LIMIT 100;`. The query result is displayed in a table with 7 columns: `id`, `name`, `description`, `firstname`, and `lastname`.

	id	name	description	firstname	lastname
1	1	CHAT	CHAT essentials	Jan	Novák
2	2	AI	AI principles	Petr	Kolečko
3	3	IoT	IoT essentials	Iva	Horáková
4	4	Wearables	Wearables for experts	Petr	Kolečko
5	5	VoIP	VoIP essentials	Marta	Kašková

Obrázek 34 - zobrazení výběru lektorů kurzů pomocí SQL dotazu v rozšíření MySQL ve VS Code

Následující ukázka zobrazuje výběr kurzu s `id=4` a jeho absolventů pomocí SQL dotazu v rozšíření MySQL ve VS Code.

The screenshot shows the Visual Studio Code interface with a MySQL database connection. The left sidebar displays the database structure, including tables like `frekventant`, `frekventant_has_kurz`, `kurz`, and `lektor`. The main editor displays a SQL query: `SELECT kurz.*, CONCAT(frekventant.lastname, " ", frekventant.firstname) AS student FROM kurz JOIN frekventant_has_kurz ON kurz.id = frekventant_has_kurz.id_kurz AND kurz.id = 4 JOIN frekventant ON frekventant.id = frekventant_has_kurz.id_frekwentant LIMIT 100;`. The query result is displayed in a table with 7 columns: `id`, `name`, `description`, `id_lektor`, and `student`.

	id	name	description	id_lektor	student
1	4	Wearables	Wearables for experts	2	Malá Ludmila
2	4	Wearables	Wearables for experts	2	Bobek Michal

Obrázek 35 - výběr kurzu s `id=4` a jeho absolventů pomocí SQL dotazu v rozšíření MySQL ve VS Code

Výhodou využití rozšíření MySQL ve VS Code je možnost vývoje aplikací bez nutnosti používání různých rozhraní a vývojových prostředí (All-in-One).