Základy relačních databází, jejich využití v programování webu

Co se v modulu dozvíte?

- Co je databáze a k čemu ji využít
- Relační databáze a jejich prvky
- Návrh a normalizace databáze
- SQL a základní dotazy

Co je databáze a k čemu ji využít?

Databázi si lze představit jako místo, kam ukládáme potřebná data. Velmi jednoduchým příkladem může být knihovna nebo kartotéka, kde jsou data uložena podle určitého systému. Pokud víme, jak tento systém použít, otevírají se nám možnosti, jak efektivně vyhledat data, jež zrovna požadujeme.

Databáze a její databázový systém slouží tedy k definici dat, která mají být ukládána. Řeší vztahy mezi těmito daty, způsob, jak je k nim přistupováno a jaké operace je s daty možné realizovat. V neposlední řadě pak funguje jako systém pro řízení přístupu k informacím – řeší oprávnění pro manipulaci s daty a správu uživatelů, jež mají k datům přístup.

Využití některého z databázových systémů jako úložiště dat a jeho propojení s programovanou aplikací je tak velmi nasnadě. Použití takového řešení je efektivnější, než vytvářet vlastní způsob ukládání dat s dostatečným zabezpečením a snadným přístupem k požadovaným informacím.

Relační databáze a jejich prvky

Relační databáze se hojně využívají v aplikacích z důvodu své dobré pochopitelnosti a jednoduchosti. Mezi neznámější zástupce relačních databázových systémů patří MySQL, Oracle, Postgress, MS SQL a další. Pro webové aplikace menšího rozsahu se nejčastěji používá databáze MySQL ve spojení s PHP.

Tabulky

Základem relačních databází jsou databázové tabulky, které jsou na sebe určitým způsobem závislé – existuje mezi nimi jistá logická vazba (relace). **Tabulky** jsou též nazývány **entitami** – jsou chápány jako prvek reálného světa (např. zaměstnanec, školní předmět, vozidlo, atd.).

Každá tabulka je tvořena **sloupci a řádky**, přičemž sloupce reprezentují vlastnosti této entity (pro příklad zaměstnance např. plat, datum nástupu, rodné číslo,...). Tyto vlastnosti se taktéž nazývají jako **atributy**. Každý sloupec musí mít jedinečný název a určený datový typ podle toho, jaká data jsou v něm uložena (číslo, text, logická hodnota, atd.).

Řádky tabulky reprezentují samotné záznamy v databázové tabulce - jeden řádek reprezentuje např. jednoho zaměstnance s hodnotami daných atributů (výší platu, rodným číslem,...). Každý řádek by měl mít svůj jedinečný identifikátor, podle kterého bude možné určit příslušný záznam - klíč.

id_zamestnance	prijmeni	jmeno	plat	datum_nastupu
1	Novák	Adam	32000	2013-05-02
2	Nová	Jana	45000	2013-06-11

Primární a cizí klíče

Primární klíče slouží jako jednoznačný identifikátor záznamu (řádku) tabulky. Jedná se o jeden nebo kombinaci sloupců, aby byla zaručena unikátnost klíče. Obvykle se jedná o číselné řady s tím, že každý záznam dostává číslo o jednotku vyšší, než ten předchozí. Při návrhu tabulky lze pro tuto funkcionalitu použít parametr `AUTO INCREMENT`, který zajistí automatické číslování.

Cizí klíče slouží k vyjádření vztahů (relací) mezi jednotlivými tabulkami. Jedná se o atribut nebo skupinu atributů, které umožní identifikovat, které záznamy z různých tabulek spolu souvisejí. Tímto způsobem se tabulky vzájemně propojují.

Základní datové typy

Každý atribut tabulky je určen datovým typem. Mezi ty nejzákladnější patří:

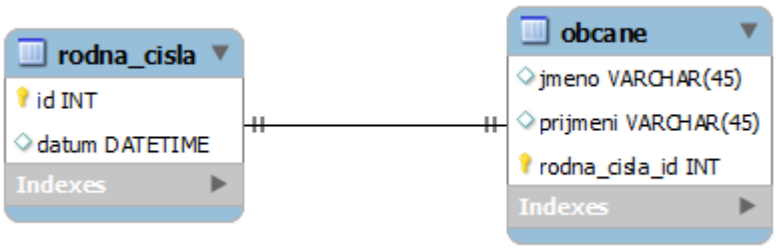
- Integer (INT) – celé číslo
- Varchar (VARCHAR) – krátký řetězec do 255 znaků
- Text (TEXT) – delší text
- Boolean (BOOLEAN) – logická hodnota
- Date (DATE) – datum ve formátu YYYY-MM-DD

Vazby mezi tabulkami

Vazba mezi tabulkami může být realizována několika způsoby. Záleží, jak jsou informace v jedné tabulce vztaženy k informacím v jiné tabulce.

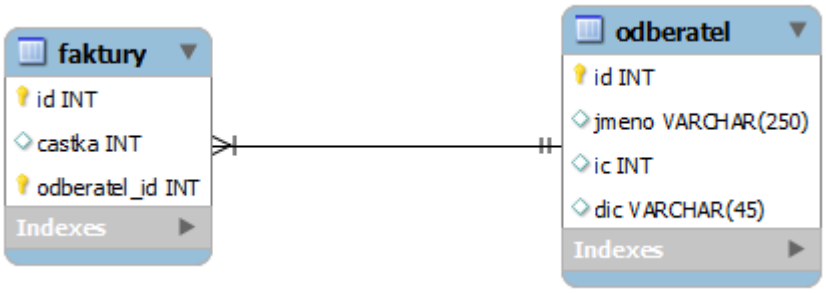
Vazba 1:1

Jedná se o spojení, kdy jedné položce v první tabulce odpovídá jedna položka v druhé tabulce. Příkladem může být tabulka Občan a tabulka Rodná čísla, kdy jednomu záznamu v tabulce Občan odpovídá jeden záznam v tabulce Rodné číslo.



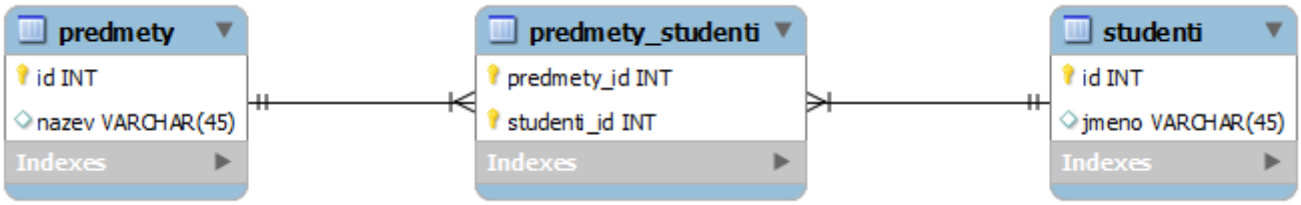
Vazba 1:n

Jedné položce v první tabulce odpovídá n-položek v druhé tabulce. Jako příklad lze uvést tabulku Odběratel a tabulku Faktura, kde faktura může mít pouze jednoho odběratele, ale odběratel může mít n-faktur.



Vazba m:n

m-položkám v první tabulce odpovídá n-položek v druhé tabulce. Např. tabulka Student a tabulka Předmět, kde student může studovat více předmětů a zároveň předmět může mít více studentů. Podobným příkladem může být ještě např. vztah mezi autorem a knihou – kniha může mít více autorů a autor může napsat více knih.



Návrh a normalizace databáze

Při vytváření aplikace je jedním z nejdůležitějších kroků správný návrh systému pro ukládání dat – vytvoření datové struktury, ve které budou uložena data, se kterými má aplikace pracovat. Pro uložení číselných, textových nebo logických dat se nejlépe hodí využití některého databázového systému z důvodů, jež jsou popsány výše. Pro vytvoření webové aplikace budeme používat databázi MySQL, která je běžnou součástí webových hostingů i instalačních balíčků pro práci s PHP na lokálním počítači (viz. kapitoly o PHP). Optimálním návrhům databáze se věnuje mnoho odborné literatury. Při návrhu běžné webové aplikace však obvykle stačí použít logické myšlení a dodržovat několik zásad.

Účel databáze

Prvním krokem by mělo být sepsání účelu, pro který databázi vytváříme. Především se jedná o to, k čemu bude databáze sloužit, jak a kdy ji bude aplikace využívat. Popis by měl posléze sloužit k identifikaci informací, které chceme v databázi zaznamenat. Pro každou základní funkcionalitu aplikace by měla být uvedena specifikace požadavků na ukládaná data. Např.: Při vypůjčení knihy aplikace využívá data o čtenáři a vypůjčované knize.

Identifikace informací a jejich atributů

Druhým krokem při návrhu databáze by mělo být shromáždění všech informací, které chceme zaznamenat. Identifikujeme dokumenty z běžné reality a vytváříme seznam jednotlivých typů informací, které tyto dokumenty obsahují. Příkladem může být např. zmíněná knihovna. Ta obsahuje knihy, časopisy, noviny..., přičemž každý tento druh položky má své atributy, které budeme chtít evidovat (autor, umístění, počet stran, popis,...). Účelem této fáze není přesně identifikovat atributy jednotlivých položek, ale vytvořit seznam údajů, jež chceme evidovat.

Rozdělení informací do tabulek

V této fázi je třeba určit entity databáze – názvy jednotlivých tabulek. Je vhodné začít u tabulek, které vyplývají z předchozí fáze a jsou určitým odrazem reality. Např. pro vytvoření jednoduché databáze e-shopu se může jednat o rozdělení na tabulky Zákazníci, Výrobky, Dodavatelé a Objednávky. Tyto tabulky potom budou obsahovat jednotlivé atributy, které je třeba k nim přidělit podle toho, co všechno chceme evidovat nebo co bude aplikace pro své fungování potřebovat. Zatím neřešíme jednotlivé vztahy, ale pouze vytváříme seznam atributů jednotlivých tabulek. Např.:

Zákazníci

- Jméno
- Adresa
- E-mail
- Telefon
- ...

Výrobky

- Název
- Cena
- Dodavatel- ...

Dodavatelé

- Jméno firmy
- Kontaktní osoba
- Telefon
- E-mail
- ...

Objednávky

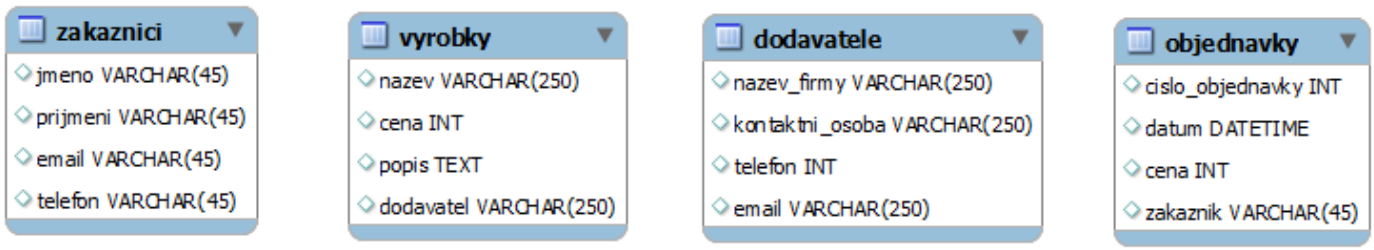
- číslo objednávky
- datum
- celková cena- ...

Myslete na to, aby se informace v tabulkách neobjevovaly duplicitně, jelikož to vede k nekonzistentnosti dat. Nastanou pak problémy při úpravě takového údaje a promítnutí změny na dalších místech. V jedné tabulce evidujte pouze atributy, které se k dané entitě vztahují.

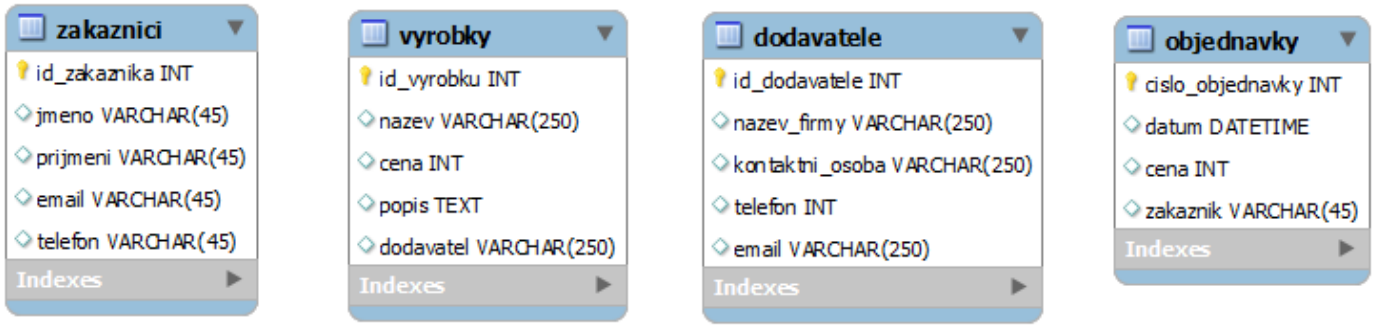
Převod jednotlivých informací do sloupců a určení primárních klíčů

Jednotlivé atributy budou reprezentovat sloupce tabulky. Nyní je potřeba určit, jak budou sloupce (atributy) pojmenovány a jaké sloupce doopravdy potřebujeme. U jednotlivých sloupců je pak zapotřebí určit datový typ pro vkládanou hodnotu.

Data se snažte do atributů rozdělovat do co nejmenších logických celků. Např. v tabulce Zákazníci je vhodné rozdělit atribut jméno na dva samostatné atributy – jméno a příjmení. Stejně tak atribut adresa by měl být rozdělen na atribut ulice, město, PSČ, případně atribut země.



V další fázi určete unikátní identifikátory jednotlivých tabulek – primární klíče. Většinou se jedná o unikátní číselnou hodnotu, která reprezentuje právě jeden řádek tabulky. Primární klíče pak poslouží ve spojení s cizími klíči k vytvoření vztahů (relací) mezi jednotlivými tabulkami. Primární klíč musí vždy obsahovat hodnotu a zvolit byste měli takový primární klíč, jehož hodnota se nebude měnit. Pokud tabulka neobsahuje vhodný sloupec/kombinace sloupců pro primární klíč (u tabulky Objednávky může být primárním klíčem sloupec číslo objednávky), je možné vytvořit nový sloupec – např. id_zakaznika, což bude unikátní číselná hodnota, reprezentující daného zákazníka.



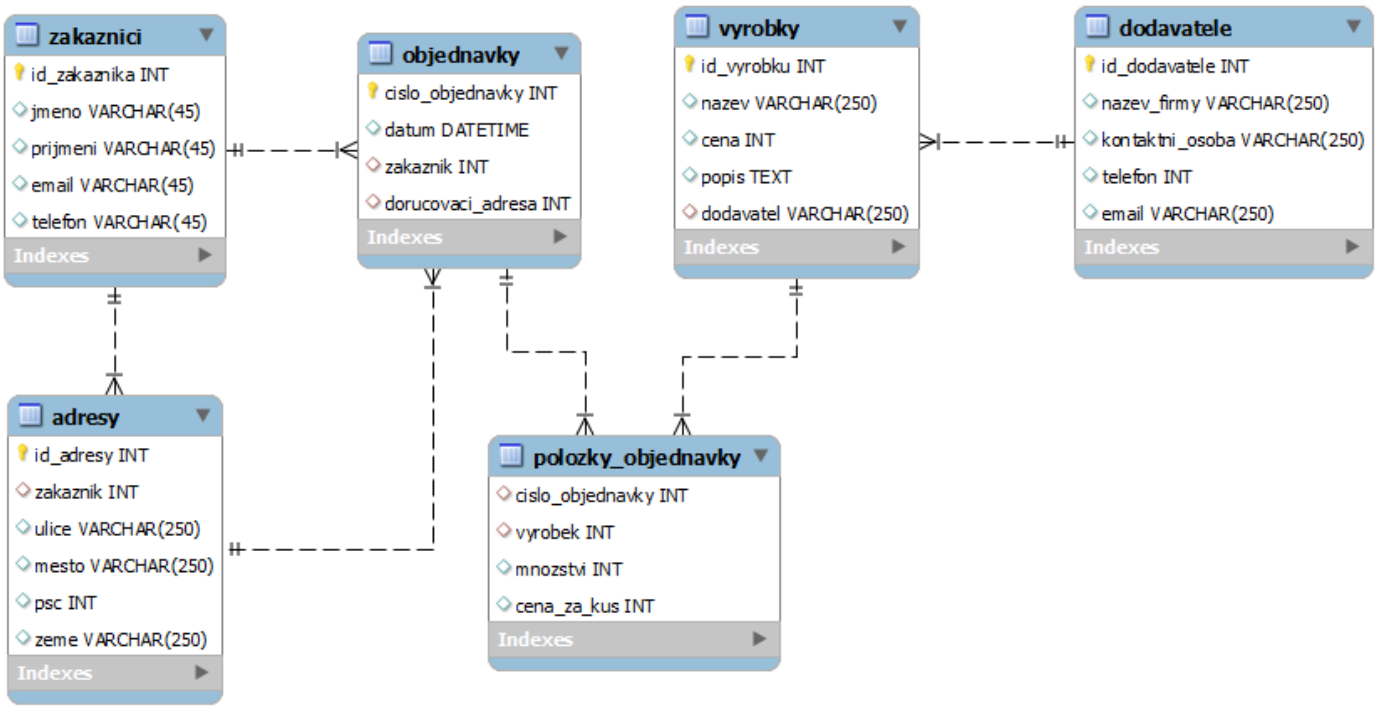
Pro návrh databáze je možné využít speciálního SW – např. programu MySQL Workbench (http://www.mysql.com/products/workbench/).

Vytvoření vztahů (relací) mezi tabulkami

Nyní je potřeba tabulky vzájemně propojit relacemi. Je třeba si uvědomit vzájemné vazby atributů z různých tabulek a jejich vzájemnou závislost (typ vazby).

Nejčastějším případem je typ vazby 1:n. Při zjištění, že náš databázový model obsahuje vazbu n:m, bude třeba vytvořit další pomocnou tabulku, jejímž obsahem budou primární klíče z dvou původních tabulek. U relací typu 1:1 zvažte, není-li výhodnější sloučit danou entitu do jedné tabulky.

V našem případě jsme propojili tabulky pomocí primárních a cizích klíčů následovně. Vždy je primární klíč jedné tabulky cizím klíčem závislé tabulky.



Např.: Primární klíč tabulky Zákazníci (id_zakaznika) je cizím klíčem tabulky Objednávky (atribut zakaznik). To znamená, že u záznamu v tabulce Objednávky bude ve sloupci zakaznik vždy vyplněna číselná hodnota, jež bude odkazovat na záznam v tabulce Zákazníci. Aplikace tak např. při zobrazení objednávky bude moci získávat aktuální data z tabulky Zákazníci (jméno, email, telefon,...).

Jelikož by byl vztah mezi tabulkami Objednávky a Výrobky typu n:m, vytvořili jsme pomocnou tabulku Položky objednávky, ve které budeme zaznamenávat jednotlivé položky všech objednávek. Příslušnost výrobku k dané objednávce je zajištěna cizími klíči, jež se vztahují k primárním klíčům tabulek Objednávky a Výrobky. U objednávek tak nemusíme evidovat celkovou cenu, ale cena bude aplikací vypočtena na základě obsahu objednávky právě z tabulky Položky objednávky. Zde naopak cenu za kus uvedeme, jelikož se v průběhu času může cena výrobku změnit, což by mělo nechtěný dopad na cenu celé vyřízené objednávky.

Doplnili jsme i tabulku Adresy, jelikož zákazník může uvést při registraci adresu fakturační nebo doručovací. Identifikátor adresy pro doručení pak evidujeme v tabulce Objednávky v atributu dorucovaci_adresa. Stejně tak je možné přidat atribut pro adresu fakturační.

Úpravy návrhu

Po vytvoření vzájemných vazeb je vhodné databázi naplnit zkušebními daty a pomocí dotazů zkontrolovat, zda databáze plní svůj účel. Pravděpodobně budou potřeba některé úpravy – např. přidání dalšího sloupce tabulky (potřebujete evidovat další atribut) nebo že je např. potřeba rozdělit tabulku na dvě, abyste odstranili duplicitní údaje nebo umožnili lepší přístup a vyhledávání v atributech. Může tedy např. nastat situace, kdy:

Zapomněli jste vložit nějaký sloupec

Přidáte atributy do jedné z tabulek nebo vytvoříte tabulku novou. Zkontrolujte však, zda nelze údaj získat prostým výpočtem z atributů, které již v tabulkách máte.

Některé sloupce přebývají

Zjistili jste, že některé atributy v tabulce lze získat výpočtem nebo je nevyužijete. Odstraňte je proto z tabulky. Jen zkontrolujte, zda na tomto atributu není vytvořena závislost.

Duplicitní informace

Pokud zjistíte, že zadáváte do tabulky stále duplicitní informace, bude třeba tabulku rozdělit do dvou tabulek, které budou propojeny vazbou typu 1:n.

Navrhli jste tabulku s množstvím atributů, ale minimem záznamů

Zvažte, zda návrh tabulky je optimální a snažte se navrhnout tabulku tak, aby obsahovala méně sloupců, ale více záznamů (řádků).

Rozdělení na nejmenší celky

Pokud potřebujete, aby bylo možné třídit záznamy podle určitého atributu, ujistěte se, že je umístěn v tabulce samostatně a jeho hodnota nabývá pouze hodnotu požadovanou k vyhledání/třídění. Příkladem je rozdělení atributu jméno na dva atributy – jméno a přímení, aby bylo možné v tabulce vyhledat příslušné příjmení.

Přidělení atributu

Ujistěte se, že všechny atributy (sloupce) v tabulce souvisí s entitou, kterou tabulka zastupuje. Příklad špatného umístění atributu - atribut datum_narozeni v tabulce Adresa.

Kompletnost relací

Zkontrolujte závislost jednotlivých tabulek a jejich kompletnost. Relace typu 1:1 a 1:n vyžadují použití primárních a cizích klíčů (společné sloupce) a relace typu n:m pak vytvoření třetí tabulky s relacemi 1:n na původní dvě (viz. náš příklad s tabulkou Položky objednávky).

Normalizace databáze

Normalizací se rozumí proces, ve kterém se snažíme návrh databáze upravovat tak, abychom mohli lépe pracovat a manipulovat s daty, zabránit redundanci dat a zvýšit konzistenci informací napříč databází.

Pravidla, která by měla být pro určitý stupeň normalizace databáze splněna, se nazývají normální formy. Čím vyšší normální forma, tím lze říci, že je databáze podle teoretických konceptů lépe navržena a mělo by být snadnější s daty pracovat. Každá normální forma v sobě zahrnuje formy předchozí (pravidla se nabalují). V realitě ale nejde vždy o dodržení všech teoretických pravidel

- databáze by měla optimálně fungovat vzhledem k danému účelu.

Normální formy

Představíme si dva základní stupně normalizace databáze.

1.NF

Každé pole v tabulce by mělo představovat jedinečná typ informace. Všechny atributy tak mají být atomické – tedy obsahovat pouze jednu nedělitelnou hodnotu. Příkladem může být právě rozdělení atributu adresa na několik atributů (ulice, město, psč,...). Pokud bychom toto pravidlo nedodrželi, měli bychom později problém, budeme-li chtít vyhledat pouze záznamy s částí adresy (např. zákazníky s určitým PSČ).

2.NF

Druhá normální forma říká, že všechny atributy mají být závislé na primárním klíči. Nelze tedy mít v tabulce atribut, který nesouvisí s primárním klíčem tabulky. Např. nebudeme dávat atribut email zákazníka do tabulky objednávky, jelikož tento atribut přímo nesouvisí s primárním klíčem číslo_objednávky. Je pro něj třeba vytvořit tabulku Zákazníci a v ní evidovat všechny atributy, které se s touto tabulkou pojí.

SQL a základní dotazy

SQL je jazyk, který slouží ke komunikaci s databází. Pomocí dotazů v určitém tvaru lze získávat z databáze data, vkládat nové záznamy či záznamy upravovat nebo mazat, vytvářet nové databáze a databázové tabulky či nastavovat přístupová práva.

Při programování jednoduché webové aplikace narazíte především na použití SQL jazyka a MySQL databáze, přičemž pro realizaci dotazů použijete funkce, které obsahuje PHP (případně některou rozšiřující knihovnu – viz moduly PHP).

SELECT

Select je jedním z nejdůležitějších příkazů a slouží k vybírání řádků a sloupců z databáze. Jako příklad si uvedeme dotaz pro získání řádků z tabulky Zákazníci a příslušných informací ze sloupců jmeno, prijmeni a email.

```
1 SELECT jmeno, prijmeni, email FROM zakaznici;
```

Pokud bychom chtěli vybrat všechny sloupce (získat všechny informace o zákazníkovi), můžeme použít dotaz ve tvaru:

```
1 SELECT * FROM zakaznici;
```

WHERE

Toto klíčové slovo slouží k doplnění dotazu o podmínku, která blíže specifikuje konkrétní záznamy. Lze tak určit, které záznamy budou vybrány – splní podmínku pro výběr.

Pro příklad, budeme chtít vybrat všechny zákazníky, kteří se příjmením jmenují Novák nebo Nováková:

```
1 SELECT * FROM zakaznici WHERE prijmeni='Novák' OR prijmeni='Nováková';
```

Použít lze také v podmínce klíčové slovo LIKE, které slouží k porovnání daného sloupce se vzorem (musí se jednat o řetězec znaků). Výhodou je možnost použití zástupného znaku a tím pádem vyhledávání záznamů podle části řetězců.

Vyhledání zákazníků, jejichž příjmení začíná řetězcem „Novák“ a může mít libovolné pokračování – tj,. např. Nováková. Procento slouží jako zástupný znak a odpovídá libovolnému řetězci:

```
1 SELECT * FROM zakaznici WHERE prijmeni LIKE 'Novák%';
```

ORDER BY

Jak anglický překlad tohoto klíčového slova napovídá, umožňuje toto slovo v dotazu řazení navrácených řádků databází podle určitého sloupce/sloupců. Pro řazení sestupně používáme klíčové slovo DESC a pro vzestupné řazení pak slovo ASC. Příklad uvádí případ, kdy chceme získat seznam všech zákazníků s příjmením od písmene V a seřazených podle toho, jak byli postupně přidáváni – tj. od nejnovějšího zákazníka po nejstaršího:

```
SELECT * FROM zakaznici WHERE prijmeni LIKE 'V%' ORDER BY id_zakaznika DESC;
```

LIMIT

V souvislosti s použitím klíčového slova ORDER BY, které seřadí záznamy podle určitého kritéria, se používá také klíčové slovo LIMIT. Pomocí něho lze zvolit, kolik záznamů chceme z databáze dostat. Příkladem může být situace, kdy budeme chtít zjistit informace o pěti nejnovějších zákaznících:

```
SELECT * FROM zakaznici ORDER BY id_zakaznika DESC LIMIT 5;
```

JOIN

Slouží ke spojování informací z několika tabulek a výsledek zobrazí jako jedinou logickou množinu záznamů - jedná se tedy o průnik dvou či více tabulek. Např. chceme vypsat čísla objednávek a k nim vždy příjmení příslušného zákazníka:

```
SELECT Objednavky.cislo_objednavky, Zakaznici.prijmeni
FROM Objednavky JOIN Zakaznici ON Objednavky.zakaznik=Zakaznici.id_zakaznika;
```

Pro klíčové slovo JOIN existuje několik variant, které umožňují práci s množinami výsledků spojení daných tabulek.

LEFT JOIN

Zkusíme vypsat záznamy všech zákazníků (levá tabulka) a k nim čísla objednávek (přičemž zákazník nemusí mít objednávku žádnou – např. po registraci nedokončil nákup):

```
SELECT Zakaznici.prijmeni, Objednavky.id_objednavky FROM zakaznici
LEFT JOIN objednavky ON Zakaznici.id_zakaznika=Objednavky.zakaznik
ORDER BY Zakaznici.prijmeni;
```

RIGHT JOIN

Toto klíčové slovo je podobné jako LEFT JOIN, jenom vybere naopak všechna data z pravé tabulky a shodné záznamy z tabulky levé. V našem případě by byl příklad irelevantní, jelikož každá objednávka by měla již mít svého zákazníka.

INSERT

Klíčové slovo, jež slouží ke vložení nového záznamu (řádku) do databáze, resp. do příslušné tabulky. Pokud tedy budeme chtít vložit do databáze nového zákazníka, provedeme to následujícím způsobem:

```
INSERT INTO zakaznici (jmeno, prijmeni, email, telefon)
VALUES ('Jan', 'Novák', 'novak@novak.cz', '+420 721 721 721');
```

Pole id_zakaznika zde není uvedeno proto, jelikož jsme pro tento sloupec (a zároveň primární klíč) nastavili vlastnost AUTO_INCREMENT – tj. hodnota bude automaticky doplněna databází (např. o jedničku větší pořadové číslo zákazníka, než má poslední zákazník v databázi).

UPDATE

Klíčové slovo slouží pro konstrukci dotazů, jež slouží k modifikaci záznamů v dané tabulce. Určit konkrétní záznam pro modifikaci lze pomocí klíčového slova WHERE. Takto např. modifikujeme záznam s příslušným ID v tabulce Zakaznici:

```
UPDATE zakaznici SET email='novak.jan@novak.cz', telefon='721721721' WHERE id_zakaznika='52';
```

Změnu lze provést také u všech záznamů v tabulce vynecháním slova WHERE. Pokud bychom měli např. v tabulce Výrobky sloupec s názvem Sleva a chtěli bychom nastavit u všech výrobků jednotnou slevu (v procentech), SQL příkaz by vypadal následovně:

```sql
UPDATE vyrobky SET sleva='5';
```

DELETE

Slouží k odstranění záznamů (řádků) z tabulky. Např. budeme chtít smazat konkrétní objednávku podle jejího čísla:

```sql
DELETE FROM objednavky WHERE cislo_objednavky='20130101';
```

Jazyk SQL samozřejmě obsahuje mnohem větší množství příkazů a možností konstrukcí dotazů. Jejich přehled lze najít např. na stránkách W3Schools pod http://www.w3schools.com/sql/ .

Jedná se např. o dotazy pro vytváření samotných databází a tabulek (`CREATE DATABASE` a `CREATE TABLE`). Tyto dotazy však budete spíše nevědomky používat v rozhraní phpMyAdmin pro správu MySQL databáze, kde vám tato aplikace usnadní práci právě při vytváření databáze a její struktury (nemusíte ručně psát dotazy pro vytváření databází, tabulek, vkládání řádků, úpravy, mazání, atd.).

Obecně platí, že by navržená databáze měla obsahovat pevný počet tabulek a aplikace by neměla pro své fungování požadovat vytváření nových tabulek a nových databází, není-li to nezbytně nutné.

Tyto materiály vznikly v rámci projektu CZ.2.17/3.1.00/34129
Rozvoj oboru Multimédia v ekonomické praxi pro lepší uplatnění absolventů v praxi

Evropský sociální fond - Praha & EU: Investujeme do vaší budoucnosti