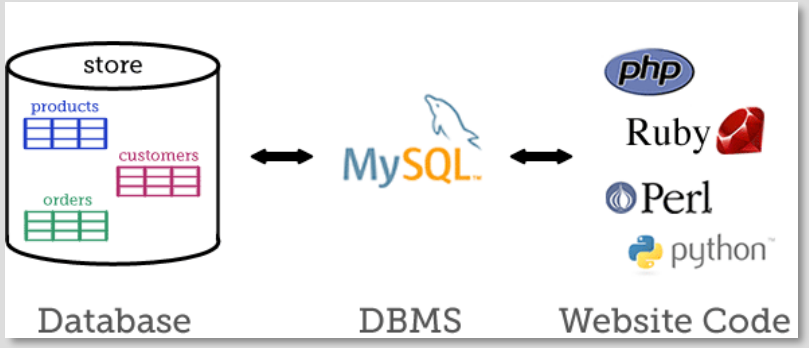
Přednáška 4: Základy databází a SQL

Databázové systémy (1/3)

Databázový systém je je kolekce navzájem vztažených **dat** spolu s **programy**, které umožňují přístup k datům. Zmíněné programy se obvykle nazývají **systém řízení řízení báze dat** (SŘBD, angl. DBMS = Database Management System). Vlastním datům se pak většinou říká **databáze**.



Činnost, kterou provádí SŘBD nad databází, se obvykle nazývá **zpracování dat** a má dvojí účel:

- organizovat velké množství dat pro **efektivní zpracování počítačem**
- organizovat interpretaci těchto dat pro **efektivní zpracování uživatelem**

Každý databázový systém musí obsahovat nástroje pro:

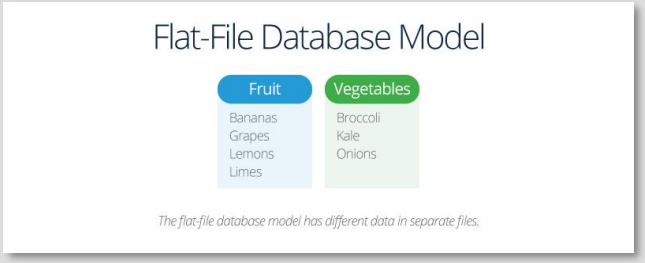
- **vytvoření, vyhledání, aktualizaci a rušení** dat
- **definici struktury** dat (tzv. Slovník dat, angl. Data Dictionary)
- **definici a zajištění integrity** (konzistence) dat (soulad mezi realitou a daty, která ji popisují)
- zajištění **fyzické a logické nezávislosti** dat (fyzická nezávislost = oddělení fyzického uložení dat od způsobu práce s nimi, logická nezávislost = změna logické struktury dat nevyžaduje úpravu dotazů, resp. programů, pracujících s daty)

Přednáška 4: Základy databází a SQL

Databázové systémy (2/3)

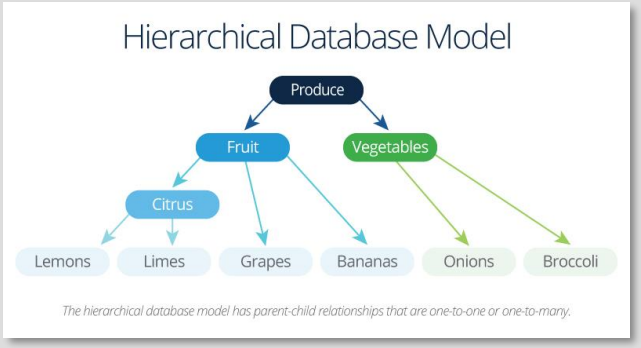
Kromě dat, uložených v databázi, je třeba mít také **popis těchto dat**. Popis dat vytváří tzv. **schéma databáze**, které popisuje manipulovatelné **objekty** v rámci databáze a **vztahy** mezi nimi. Tyto objekty jsou z hlediska abstrakce a vnitřní struktury závislé na použitém **databázovém modelu**.

Databázový model je pak specifikace popisující jak je databáze **strukturována a používána**. Mezi nejběžnější databázové modely patří:



Flat-File model (textový soubor)

Data jsou uložena v souborech, mezi soubory ale neexistují žádné vzájemné vazby.



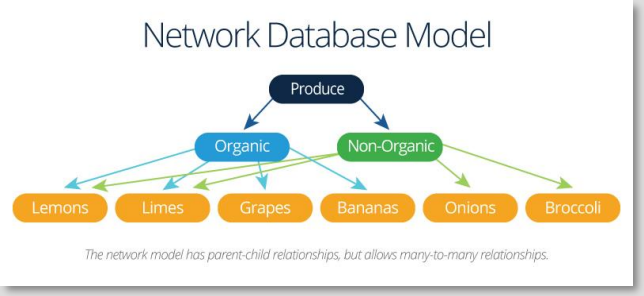
Hierarchický model

Každý prvek v modelu má pouze jediný nadřazený prvek, ale může mít více podřízených prvků. Prvek má tedy vztah k právě jednomu rodiči, ale jednomu nebo více potomkům.

Zdroj obrázků: https://www.smartsheet.com/relational-database-modeling

Přednáška 4: Základy databází a SQL

Databázové systémy (3/3)

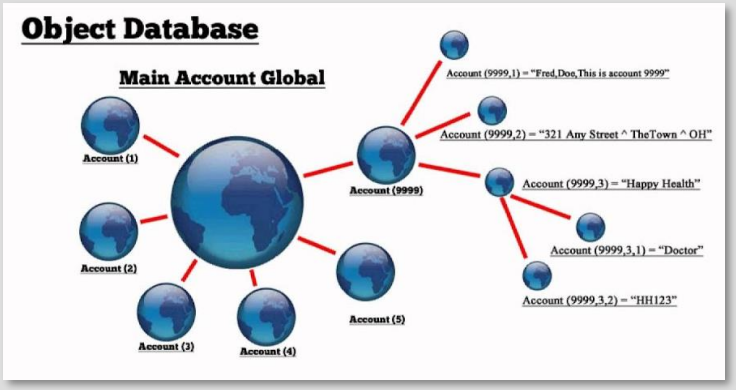


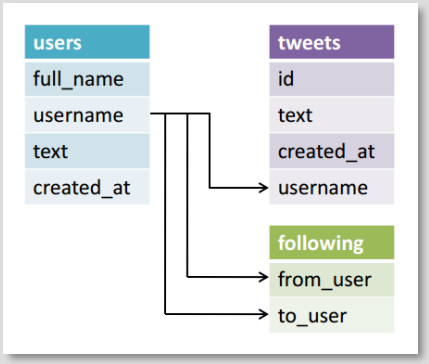
Síťový model

Je podobný hierarchickému modelu, ale na rozdíl od něj dovoluje použití m:n vazeb. Tj. prvek může mít více než jednoho rodiče.

Objektový model

je datový model založený na objektově orientovaném programování. Sdružuje metody (funkce) s objekty, které mohou využívat hierarchie tříd.





Relační model

V relačním modelu jsou data organizována v tabulkách, které se skládají z řádků a sloupců. V těchto tabulkách jsou prováděny všechny databázové operace.

Logické schéma databáze je ta část schématu databáze, kterou popisuje databázový model. Oproti tomu **fyzické schéma databáze** popisuje uložení dat na vnějších pamětech, resp. způsob přístupu k nim.

Přednáška 4: Základy databází a SQL

Návrh databáze (1/6)

Návrh databáze obvykle probíhá ve třech krocích (vytvoření a zpřesňování logického databázového schématu a následně jeho implementace):

1. Konceptuální úroveň
Na této úrovni se snažíme vytvořit popis dat **nezávisle na** jejich následném **uložení** v databázi. Tzv. konceptuální model tedy představuje **formální popis** modelované reality. Hlavními úkoly pak je nalezení entit, vtahů a atributů.

Konceptuální modely tedy používají pojmy:
- **entita** (objekt) – např. student, předmět
- **vztah** (relationship) – např. studuje
- **atribut** (vlastnost) – např. věk, rč

Mezi cíle konceptuálního modelu patří:
- **vytvořit obraz reality ve formalizované podobě** nezávislý na pozdější formě implementace
- **formalizovat požadavky uživatelů** a návrhářům poskytnout prostředek pro komunikaci s uživateli
- **vytvořit podklad pro návrh databáze**

Přednáška 4: Základy databází a SQL

Návrh databáze (2/6)

E-R a ERA diagramy
Pro formální zobrazení reality se na konceptuální úrovni návrhu používají tzv. E-R modely, resp. jejich grafické znázornění **E-R (Entity-Relationship) diagramy** (česky diagramy entit a vztahů). Pokud do modelu, resp. diagramu zavedeme také atributy dostaneme **ERA (Entity-Relationship-Attribute) diagramy**.

Pro kreslení obou typů diagramů je k dispozici velké množství **notací**, které se liší počtem značek, způsobem zobrazování vztahů, apod. Často citovaná je notace podle **Petera Chena** z roku 1976, alternativně ale vznikla tzv. notace **„Crow's Foot"** (vraní noha), která je v současnosti používaná množstvím aplikací (např. Microsoft Visio, MySQL Workbench / DBDesigner 4 a další).

Prvky v datových modelech

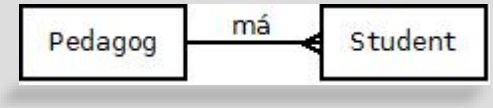
Entita
Jedná se o významný prvek ve zkoumané oblasti, např. entita **Student**. Entity se v diagramu zobrazují jako obdélníky s vepsaným názvem entity.

Atribut
Atributy entity **Student** mohou být např. **jméno**, **ročník**, apod. Není nutné je přímo vyznačovat, postačí je uvést v textovém komentáři k diagramu.

Přednáška 4: Základy databází a SQL

Návrh databáze (3/6)

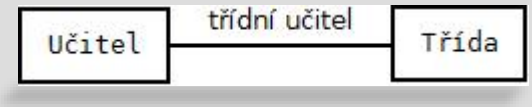
Vztah
Libovolný vztah, ve kterém se mohou nacházet **dvě nebo více** entit. Např. vztah **„Pedagog má studenty"** je vyjádřením vztahu ***má*** entit **Pedagog** a **Student**. Vztah se vyznačuje plnou čarou spojující entity, které vystupují v daném vztahu. Je rovněž vhodné vztah pojmenovat.



Kardinalita vztahu
Smyslem kardinality vztahu je vyjádřit **počet výskytů objektů** obou zůčastněných entit. V našem případě víme, že pedagog je pouze jeden, ale má (např. v rámci předmětu) více studentů. Jedná se proto o příklad vztahu **1:n** a graficky jej znázorníme tzv. **„vidličkou"** na straně n-výskytů objektů entity.

Typy kardinalit vztahů

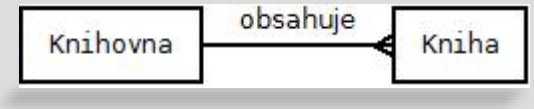
1:1
Na **obou stranách** vztahu vystupuje **výhradně jeden** objekt entity. V diagramech se vyskytují **zřídka**, příkladem může být vztah ***třídní učitel*** mezi entitou **Učitel** a entitou **Třída** nebo třeba **Student** – **Místo k sezení**, apod.



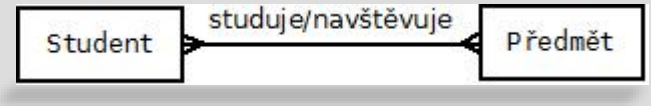
Přednáška 4: Základy databází a SQL

Návrh databáze (4/6)

1:n
Na jedné straně vystupuje **jediný objekt**, který je **ve vztahu s více objekty** entity na straně druhé. Tento vztah se v praxi vyskytuje velmi často (např. vztah **Pedagog - Student**, **Knihovna - Kniha**, apod.)



m:n
Specifický typ vztahu, kde se **na obou stranách** může vyskytovat **více objektů** každé z entit. Příkladem může být vztah **Student - Předmět**, kde jeden předmět může navštěvovat více studentů a zároveň jeden student může studovat více předmětů. V praxi se pak takovýto vztah řeší **dekompozicí** na dva vztahy 1:n.

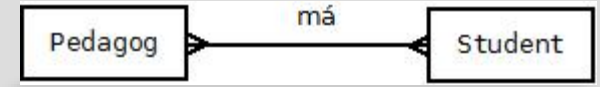


Parcialita vztahu
Kromě kardinality vztahu můžeme také zvolit, jestli je vztah **volitelný** nebo **povinný**, tj. tzv. **parcialitu vztahu**. Lze si položit např. otázky typu: „Musí mít každý pedagog studenty?" nebo „Může existovat kniha, která nepatří do žádné knihovny?", apod.
Parcialita vztahu se v diagramech vyjadřuje **kroužkem** na straně entity, jejíž objekty nemusí existovat.

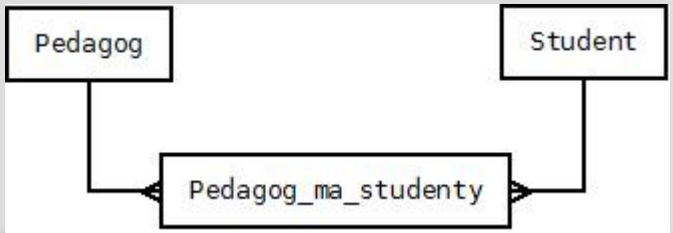
Přednáška 4: Základy databází a SQL

Návrh databáze (5/6)

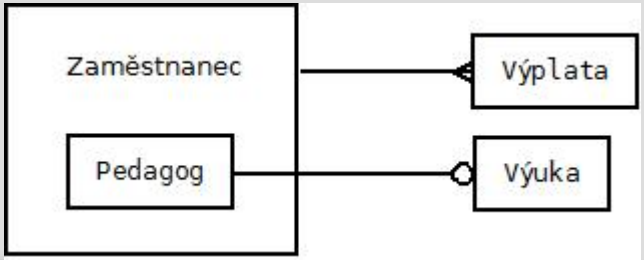
Dekompozice vztahů m:n
Tímto pojmem máme na mysli vytvoření nové, tzv. **vazební entity**, která bude mít k původním entitám vztah **1:n**.



Nově vzniklá entita **Pedagog_ma_studenty** vyjadřuje fakt, že student studuje u více pedagogů a zároveň, že pedagog může mít najednou více studentů.



Generalizace a specializace
Generalizací rozumíme **zobecnění** některých vlastností různých entit, výsledkem kterého je splynutí těchto entit v jednu. Např. pedagog, technik a uklízečka jsou zaměstnanci univerzity. Naopak **specializací** je míněn **opačný** postup.



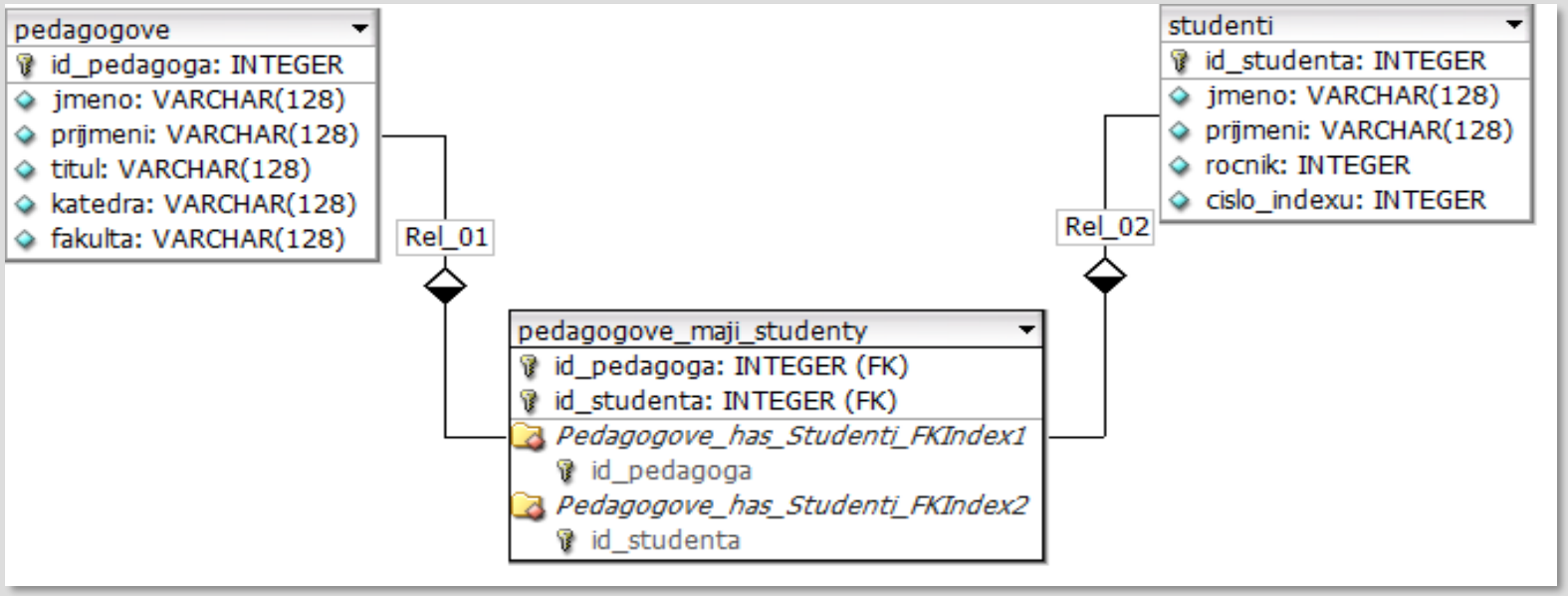
Notace specializace

Přednáška 4: Základy databází a SQL

Návrh databáze (6/6)

2. Logická úroveň

Na této úrovni se v relačních databázích používá tzv. relační schéma. Toto schéma již obsahuje podrobné popisy jednotlivých tabulek včetně jejich sloupců, tabulky obsahují primární klíče a rovněž cizí klíče, jako odkazy na primární klíče jiných tabulek.



3. Implementační úroveň

Na této úrovni již vybíráme konkrétní databázový systém pro realizaci návrhu databáze. Např. MySQL, PostgreSQL, Oracle, MS SQL Server, apod.

Přednáška 4: Základy databází a SQL

Klíče

Primární klíč je hodnota sloupce nebo kombinace hodnot více sloupců najednou, která **jednoznačně identifikuje každý záznam v tabulce**. Žádný primární klíč nesmí obsahovat hodnotu NULL a každá tabulka by měla mít definován právě jeden primární klíč.

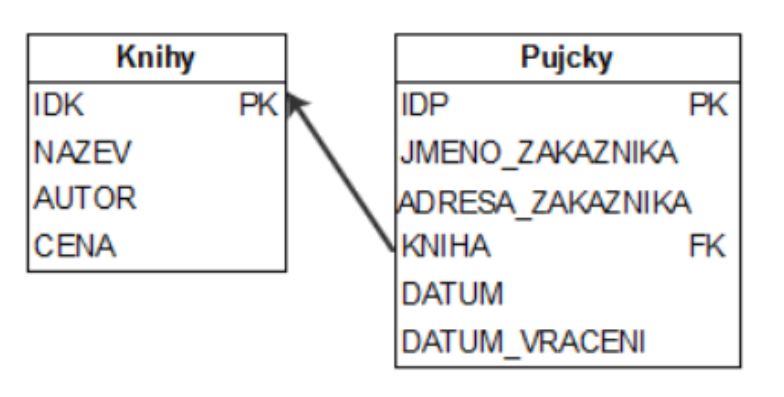
Databázový systém by měl být navržen a udržován tak, aby se primární klíč záznamu nemusel nikdy měnit.

Cizí klíč je hodnota sloupce nebo kombinace hodnot více sloupců najednou, která **jednoznačně identifikuje řádek jiné tabulky**.

Jinak řečeno, cizí klíč je definován ve druhé tabulce, ale odkazuje na primární klíč v první tabulce.

Přidáním cizího klíče vytváříme tzv. **integritní omezení**, které do tabulky umožní vložit pouze povolené hodnoty. Také určuje, co se děje se souvisejícími daty, když je spojený záznam smazán nebo upraven.

Omezení cizích klíčů nabízí způsob pro udržení referenční integrity databáze. V **MySQL** databázi lze toto omezení provádět pouze nad tabulkami, které používají úložiště **InnoDB**.



Přednáška 4: Základy databází a SQL

Normální formy (1/3)

Normální formy tabulek se používají v **relačním modelu**. Jedná se o souhrn pravidel, která umožňují postupně zkvalitnit implementaci modelu. Při procesu tzv. **normalizace** tedy dochází k postupnému **odstraňování nedostatků** tabulek. Při dosažení určité úrovně říkáme, že se tabulka nachází v určité normální formě. Normálních forem je celkem pět, v praxi se ale obvykle používají první tři:

1NF (první normální forma): Všechny sloupce (atributy) tabulky jsou **atomické**, tj. již dále nedělitelné. Příklad:

jmeno	**prijmeni**	adresa
Josef	Novák	Jablonecká 99, Liberec, 46000

Tabulka v 1NF

jmeno	**prijmeni**	ulice	mesto	psc
Josef	Novák	Jablonecká 99	Liberec	46000

2NF (druhá normální forma): Tabulka splňuje 2NF, právě když splňuje 1NF a navíc **každý sloupec**, který není primárním klíčem je **na primárním klíči úplně závislý**. To znamená, že se nesmí v řádku tabulky objevit položka, která by byla závislá jen na části primárního klíče. Příklad:

jmeno	**kod_predmetu**	nazev_predmetu
Novák	IZI	Internet a zdravotnicka informatika

Problematiku 2NF musíme řešit v případě, že tabulka má složený primární klíč, tj. např. výše, kdy je **primárním klíčem** kombinace **dvou** sloupců: **jmeno** a **kod_predmetu**.

Přednáška 4: Základy databází a SQL

Normální formy (2/3)

Problém nastává u sloupce **nazev_predmetu**, protože ten je zavislý pouze na **jedné části** primárního klíče – sloupci **kod_predmetu**. Např. při změně názvu předmětu bychom tak museli přejmenovat hodnoty ve všech řádcích sloupce nazev_predmetu. Proto tato tabulka **není v 2NF**.

jmeno	**kod_predmetu**	nazev_predmetu
Novák	IZI	Internet a zdravotnicka informatika

Řešením je tabulku **rozdělit** (dekomponovat) na dvě tabulky:

jmeno	kod_predmetu
Novák	IZI

kod_predmetu	nazev_predmetu
IZI	Internet a zdravotnicka informatika

Obě tabulky již 2NF splňují.

Přednáška 4: Základy databází a SQL

Normální formy (3/3)

3NF (třetí normální forma): Tabulka je nejméně ve 3NF, pokud je ve 2NF a neobsahuje **žádnou mimoklíčovou závislost**. Lze si to představit tak, že by existovaly dva sloupce ležící mimo klíč, u kterých by bylo snadné určit, který je příčinou a který jednoznačným následkem. Příklad:

jmeno	**kod_predmetu**	mistnost	kapacita
Novák	IZI	F-UZS1	15

Řešením je opět tabulku **rozdělit** (dekomponovat) na dvě tabulky:

jmeno	**kod_predmetu**	mistnost
Novák	IZI	F-UZS1

Obě tabulky jsou ve 3NF.

mistnost	kapacita
F-UZS1	15

Přednáška 4: Základy databází a SQL

SQL (1/6)

SQL je standardizovaný dotazovací jazyk používaný pro práci s daty v **relačních databázích**. Je to zkratka anglických slov **Structured Query Language** (strukturovaný dotazovací jazyk).

Historie SQL
V 70. letech 20. století probíhal ve firmě IBM výzkum relačních databází. Bylo nutné vytvořit sadu příkazů, kterými by se relační databáze ovládala. Vznikl proto jazyk SEQUEL (Structured English Query Language). K vývoji jazyka se následně přidaly i další firmy. V r. 1979 uvedla na trh firma Relational Software, Inc. (dnešní Oracle Corporation) svou relační databázovou platformu Oracle Database. IBM uvedla v roce 1981 nový systém SQL/DS a v roce 1983 systém DB2. Dalšími systémy byly např. Progres, Informix a SyBase. Ve všech těchto systémech se používala varianta jazyka SEQUEL, který byl přejmenován na SQL.

Vzrůstající význam relačních databází si vyžádal standardizaci jazyka pro jejich ovládání, proto byl v roce 1986 definován nový standard založený na jazyce SQL, podle roku vydání označovaný jako SQL86. Následoval pak SQL-92 (označovaný také jako SQL2). Zatím nejnovějším standardem je SQL3 (SQL-99), který reaguje na potřeby nejmodernějších databází s objektovými prvky.

SQL byl navržen pro konkrétní účel - **dotazovat se na data obsažená v relační databázi**. SQL je **deklarativní programovací jazyk**, nikoli imperativní programovací jazyk jako C nebo BASIC. Rozšíření standardního SQL však funkce procedurálního programovacího jazyka, jako jsou konstrukce řízení toku, apod., přidávají.

Přednáška 4: Základy databází a SQL

SQL (2/6)

Dotazy pomocí příkazu SELECT
Slouží k vyhledávání dat podle různých kritérií. Příkaz má mnoho variant, kompletní dokumentaci lze nalézt na serveru www.mysql.com.

Příklady:

Výpis **všech informací** z tabulky
*SELECT * FROM studenti;*

Výběr sloupců jmeno a prijmeni (tzv. projekce)
SELECT jmeno, prijmeni FROM studenti;

Omezení počtu řádků (tzv. restrikce)
*SELECT * FROM studenti WHERE kod_predmetu = „IZI";*
SELECT jmeno, prijmeni FROM studenti WHERE kod_predmetu = „IZI";

Třídění řádků na výstupu
SELECT jmeno, prijmeni FROM studenti WHERE kod_predmetu = „IZI" ORDER BY prijmeni;

Získání **souhrnných informací**
SELECT COUNT(*) FROM studenti WHERE kod_predmetu = „IZI";
SELECT MAX(hodnoceni) FROM studenti WHERE kod_predmetu = „IZI";

Přednáška 4: Základy databází a SQL

SQL (3/6)

Získání **souhrnných informací za skupinu**
Potřebujeme-li znát souhrnné informace za určitou **pomnožinu záznamů**, musíme použít klíčové slovo **GROUP BY**. Např. chceme znát počty studentů, resp. jejich nejvyšší hodnocení za každý předmět:

SELECT COUNT(*) FROM studenti GROUP BY kod_predmetu;
SELECT MAX(hodnoceni) FROM studenti GROUP BY kod_predmetu;

Spojování tabulek
Potřebujeme-li vypsat **jména studentů** a zároveň **názvy předmětů**, které studují, použijeme **spojování tabulek**. Předpokládejme, že podle 2NF máme obě informace uložené ve dvou různých tabulkách:

SELECT jmeno, prijmeni, nazev_predmetu FROM studenti JOIN predmety ON studenti.kod_predmetu = predmety.kod_predmetu;

Alternativní syntaxe:

SELECT jmeno, prijmeni, nazev_predmetu FROM studenti, predmety WHERE studenti.kod_predmetu = predmety.kod_predmetu;

Přednáška 4: Základy databází a SQL

SQL (4/6)

Přidávání řádků
Pomocí příkazu **INSERT** můžeme do existující tabulky přidat **nové řádky**. Vkládáme buď **hodnoty všech sloupců** tabulky nebo jejich **podmnožinu** (v závislosti na atributu auto_increment, resp. null), kterou uvedeme do závorky za název tabulky.

INSERT INTO pedagogove VALUES ('', 'Josef', 'Novák', 'Ing.', 'KOD', 'FT');
INSERT INTO studenti (jmeno, prijmeni) VALUES ('Josef', 'Novák');

Oprava hodnot v tabulce
Pro tento účel se používá příkaz **UPDATE**. Při jeho používání nedochází ke změně počtu řádků, ale pouze k **úpravě hodnot** v jednom či více sloupcích u jednoho či více řádků tabulky.

UPDATE pedagogove SET katedra = 'KTT' WHERE prijmení = 'Novák';
UPDATE pedagogove SET katedra = 'KTS', fakulta = 'FS' WHERE id_pedagoga = 1;

Vymazání řádků
Lze provést příkazem **DELETE** a to (v závislosti na podmínce) buď všechny nebo jen vybrané řádky.

DELETE FROM studenti;
DELETE FROM pedagogove WHERE katedra = 'KTS';

Přednáška 4: Základy databází a SQL

SQL (5/6)

Vytvoření tabulky
K tomuto účelu se používá příkaz CREATE TABLE. Kromě zadání názvu vytvářené tabulky musíme definovat názvy, datové typy sloupců a další parametry tabulky.

```
CREATE TABLE pedagogove (
  id_pedagoga int(10) unsigned NOT NULL AUTO_INCREMENT,
  jmeno varchar(128) DEFAULT NULL,
  prijmeni varchar(128) DEFAULT NULL,
  titul varchar(128) DEFAULT NULL,
  katedra varchar(128) DEFAULT NULL,
  fakulta varchar(128) DEFAULT NULL,
  PRIMARY KEY (id_pedagoga)
) ENGINE=InnoDB;
```

Nejpoužívanější datové typy v SQL

INT[(*zobrazovaci_sirka*)] – běžně velká celá čísla
- parametr **zobrazovaci_sirka** nastavuje max. zobrazovaci sirku celého čísla, do které je doplněno mezerami, příp. nulami (v případě použití parametru ZEROFILL)
- atribut **unsigned** znamená, že zadáváme číslo bez znaménka, tj. jen kladná čísla
- atribut **AUTO_INCREMENT** znamená, že pokud vložíme hodnotu NULL nebo '' (prázdný řetězec) vloží se celé číslo o jedničku vyšší než obsahuje předchozí řádek

Přednáška 4: Základy databází a SQL

SQL (6/6)

FLOAT(*cele_cislo*) – desetinné číslo s plovoucí desetinnou čárkou
- parametr **cele_cislo** udává počet platných číslic (max. 38)

VARCHAR[(*delka_retezce*)] – řetězec znaků pevně dané délky
- parametr delka_retezce nastavuje delku retezce, při jejimz prekroceni se retezec zkrati; v pripade, ze je retezec kratsi nez parametr, se mezery odstrani

TEXT – hodnota text
- povolená délka řetězce je 0 až 65535 bajtů

DATETIME - datum a čas ve formátu 'CCYY-MM-DD hh:mm:ss'

TIMESTAMP - časová značka ve formátu 'CCYY-MM-DD hh:mm:ss' (obsahuje datum i čas). Vložíte-li do sloupce, kde je nastaven TIMESTAMP, NULL, do databáze se uloží aktuální datum a čas ve formátu popsaném výše. Upravujete-li jakýkoli sloupec v řádku, kde je nastaven také TIMESTAMP, upraví se i sloupec TIMESTAMP, a to na aktuální datum a čas, kdy byla změna provedena.

Odstranění tabulky
DROP TABLE studenti;

Vyprázdnění tabulky
TRUNCATE TABLE studenti; ***nebo*** *DELETE FROM studenti;*