1 Databáze tvorba

1.1 Teorie návrh databáze – normalizace

Než začnete tvořit je potřeba vědět co a jak (třeba už víte jaké ***názvy sloupců*** raději použít ať si následně ušetříte práci při psaní SELECTU).

- Pokud jsou dvojslovné komplikovanější odkazy svislá čátka
- Uživatele_id vs id_uzivatele
- Čeština a speciální znaky

Víte v jaké ***jazykové sadě*** budete chtít mít, jak ***řadit*** už víte jak se řadí ve vaší, kde je ch?

- kde je č, ch?

Následné změny v databázi se promítnou jen do nově vytvořených tabulek > zmatek část v jenom formátu část ve druhém.

- Co funkce procedury atd, které jsou na „pozadí“ (budeme se bavit)

Chcete / musíte rozlišovat malá a velká písmena v SELECT?

Podíváme se jak vytvořit logické celky, odborně normalizovat databázi. Sice ***učebnice*** tvrdí jednu věc ***praxe*** pak provádí jinou.
Ne vše normalizuje (rozděluje) na jednotlivé části. Důležitá data raději nechává v jedné tabulce. Přeci jen propojování nemusí díky chybně zadanému SELECTU dopadnou správně.
Případně rozděluje (bezpečnost dat, práva)

1.1.1 Co je normalizace

Proces kdy, zajistíte ať se v databázi neopakují stejné údaje.
Tj. budu mít tabulku nákupy. A k výrobku do tabulky uvedu firmy, její adresu atd., když člověk.
Před normalizaci máte „velkou“ tabulku, která nebyla rozčleněna na jednotlivé logické celky.

1.1.2 Logický návrh

Databázi je potřeba **navrhnout pro koncové uživatele**: (funguje jinak než teorie)

- Zde kámen úrazu (něco co je logické pro programátory), nemusí být logické pro koncové uživatelé
- Tady se většinou hodě šetří a pak to tak dopadá SAP (nebo jiný „CMR“) už nákup samotné databáze je drahá záležitost, vlastní implementace ještě dražší a ohnutí do patřičné podoby velice nákladné.
- Proto se použije většinou standartní „šablona“
- Kdo vkládá data do databáze? Co umí Opravdu? Děla někdo s nějakým EPR-kem ve firmě? Podnikovým systémem **Ono i FB / IS je databáze**.

Co by měl logický návrh zabezpečit

- Snížit množství dat
- Eliminovat opakující se data
- Důležité pro UX co uživatel uvidí

1.1.3 Jak navrhnout?

Potřeby uživatele! Ne to co chce, ale co potřebuje. Ne mluvit jen se šéfy, ale s lidmi co to budou používat! Dozvíte se mnoho cenného a to co na začátku vypadá jako ztráta času tak vám v budoucnu ušetří spoustu čas a problému (prostě v tabulce budou dva sloupečky navíc a přidá se ještě jedna tabulka).

Pár otázek na uživatelé (klidně si je pak dále doplňujte):

- **PROČ?** JAK?
- Jaké data používáte?
- Co potřebujete vidět (souhrny, výpočty, přehledy)
- Kdo bude používat (práva, bezpečnost)
- Jak data seskupená, existují nějaké logické celky?
- Priority, co děláte nejčastěji?
    - Ke kterým datům přistupujete
- Jak dlouho do historie potřebujete zasahovat.
- Jazyk(y)?
- Firemní procesy, jak „data“ informace putují.

1.1.4 Redudance - neopakovat

Neměla by být duplikace dat - někdy je nutná (bezpečnost…).
Primárně netřeba - data se zbytečně opakují, zabírají prostor, musím je kontrolovat a když se změní jedny musím změnit i ty druhé.
Když budu mít 2x zadanou adresu, které je ta pravá?

- Možná pokud v bance máte účet i úvěr přestěhujete se změníte jednu nezmění se obě!!!
    - Hypotéka
    - BU
    - Zmíním ještě u tvorby tabulek.

Pojďme se podívat co kromě redundance ovlivňuje návrh:

1.1.5 Co ovlivňuje návrh?

Správný návrh důležitý! Ušetří hodně problému. **Nedáte napoprvé**! Potřeba znalost, praxe.
Tip: Koukat jak a proč jsou tak vytvořené cizí databáze. Ukážeme si prakticky jak na to (sakila, jiné?).
Faktory, které je třeba brát v úvahu:

- Bezpečnost
- Typ databáze (umí cizí klíče)
- Normalizace … za chvílí podrobněji
- Rychlost operací
- Podporuje operace (MySQL neumí operátory MINUS….)
- Umí databáze
    - INDEXY
    - Dočasné tabulky
    - Pohledy
    - ….
- Prostor na disku, umístění (mnohdy servery na jiném kontinentě) … čekat 10 min na výsledek?
- **Kdo bude s databázi pracovat** … málo se zmiňuje, ale vychytávky (vyhledávaní podle regulárních výrazů) mnoho lidí nenadchne, ale 3 tlačítka zaujmou….
    - Programátoři

- Uživatelé

1.2 Normální formy

Způsob do jaké hloubky je databáze normalizovaná … neboli ať se údaje neopakují. Jsou 3 nejčastější formy normalizace:

- První
- Druhá
- Třetí

Dokud není předchozí normalizace provedena nelze provádět následující, tj. nelze začít a provést jen třetí

1.2.1 Data

Máme k dispozici údaje o zaměstnancích a zákaznicích. Chceme sbírat údaje jako jejich jméno…

Sloupce	zaměstnanec	Zákazník
Jmeno	x	X
Prijmeni	x	X
adresa_ulice	x	X
aresa_mesto	x	X
adresa_PSC	x	X
Telefon	x	X
Email	x	X
zam_pozice	X	
zam_oddeleni	X	
zam_plat	X	
zam_nastup	X	
Objednavka		X
Poznámka		x

Co mi chybí? Jak mohu získat
Typ zaměstnanec/ zákazník

1.2.2 První normální forma

Cílem je rozdělit data na logické tabulky. Vidím, že v jedné velké tabulce budou zbytečně nevyplněná místa a mnoho informaci se bude opakovat.
Takže v prvním kroku na logické celky s **primárním klíčem**.

- Primární klíč – sloupec díky kterému je každá řádek (záznam) jedinečný. Tj. UČO, rodné číslo, … ID zaměstnance, žádné UČO se neopakuje

Jak bude vypadat?

zaměstnanec
ID_zamestnanec
jmeno
prijmeni
adresa_ulice
aresa_mesto
adresa_PSC
telefon
email
zam_pozice
zam_oddeleni
zam_plat
zam_nastup

zákazník
ID_zakaznik
jmeno
prijmeni
adresa_ulice
aresa_mesto
adresa_PSC
telefon
email
objednavka
poznamka

1.2.3 Druhá normální formy

Data částečně závislá na primárním klíči a uložit je do samostatných tabulek. Tj. v právní bych neustále psal pozici oddělení > při změně pozice (přejmenování pozice). U druhé objednávce bych zopakoval adresu atd. Musel bych vyplnit…
Tabulky mají primární klíč. Tj. sloupec podle které můžeme spojovat (což už umíme).

zaměstnanec
ID_zamestnanec
jmeno
prijmeni
adresa_ulice
aresa_mesto
adresa_PSC
telefon
email

zákazník
ID_zakaznik
jmeno
prijmeni
adresa_ulice
aresa_mesto
adresa_PSC
telefon
email

zaměstnanec_pozice
ID_zamestnanec
zam_pozice
zam_oddeleni
zam_plat
zam_nastup

zákazník
ID_zakaznik
objednavka
poznamka

1.2.4 Třetí normální forma

Odstranit data která nejsou závislá na primárními klíči. Která to mohou podle vás být?

Pojmenování tabulek je na vás. Už víte, že mezera může zlobit, stejně tak jako české znaky (ukážeme si jak nastavit české prostředí ve tvorbě).
Buď si zvolíte své, nebo dostanete příkazem.
PSČ, co pozice? Neopakuje se?
Co oddělení? Neopakuje se?
Nedochází k přejmenovávání oddělení?
U tvorby tabulek se budeme i o cizích klíčích.
Takže, můžeme rozdělit takto:

zaměstnanec
ID_zamestnanec
jmeno
prijmeni
adresa_PSC
telefon
email

zákazník
ID_zakaznik
jmeno
prijmeni
adresa_PSC
telefon
email

Adresy
adresa_PSC
adresa_ulice
aresa_mesto

zaměstnanec_HR
ID_zamestnanec
ID_pozice
ID_oddělení
zam_plat
zam_nastup

objednavky
ID_zakaznik
objednavka
poznamka

pozice_HR
ID_pozice
zam_pozice

oddeleni_HR
ID_oddělení

zam_oddeleni

- Co se stane (budeme chtít ať se stane, pokud smažeme oddělení)? Budeme chtít?
- Co se stane pokud zadáme odkaz na oddělení které neexistuje?

Neboli referenční integrita…

1.2.5 Referenční integrita

Aneb záruka konzistence dat. Že si data odpovídají. Předchozí případ, co by se stalo pokud někdo smaže oddělení? Pár zaměstnanců nebude nikam patřit ☹ Nebo budu přidávat zaměstnance a dám odkaz na neexitující ID oddělení?
Tj. potřebuji kontrolovat, zda záznam v tabulce existuje. Integrita se řídí prostřednictvím klíčů. Primárních ten už víme, je unikátní v dané tabulce a cizích.

- Cizí klíč – neboli se odkážeme do jiné tabulky na dané pole v záznamu.
- Vztah může být 1:1

Nebo 1:1

zamc		zamc_HR
ID_zmc	1 : 1	**ID_zmc**
jmeno		zam_pozice
…		…

zamc	
ID_zmc	**jmeno**
1	Iva
2	Ivo
3	Eva

zamc_HR	
ID_zmc	**zam_pozice**
1	IT
2	HR
3	PR

Nebo 1:N

Objednavky		Produkty
ID_Obj		**ID_Produkt**
ID_Produkt	1 : N	Nazev
		…

Objednavky	
ID_Obj	ID_Produkt
1	1
2	1
3	1
4	2
5	2
6	3

Produkty	
ID_Produkt	**Nazev**
1	Auto
2	kolo
3	lod

Nebo N:N

Kniha		Čtenář
ID_kniha	N : N	**ID_ctenar**
pozn		pozn

Knihy		Propojeni			Produkty
ID_kniha	1 : N	**ID_Kniha**	**ID_ctenar**	N : 1	**ID_ctenar**
pozn					pozn

1.2.6 Výhody normalizace

Normalizace má tyto výhody:

- Lepší uspořádání
- Menší databáze (data se neopakují)
- Konzistence – logičnost
- Lepší možnost zabezpečení

1.2.7 Nevýhody normalizace

Mají také nevýhody

- Nižší výkon
- Pomalejší (spojovací dotazy jsou logicky pomalejší)
- Vnořené ještě pomalejší

1.2.8 Denormalizace

Umím provádět normalizaci a mnohdy se z normalizované databáze se denormalizuje (jdeme „zpět“, proč? Chceme ať se data v tabulce opakují, proč?

- **Výkon** … spojovací dotazy jsou pomalejší, prostor už není tak drahý.
- **Bezpečnost** – chcete zabezpečit ať uživatele tabulek nevidí na data mezi sebou.
    - Neboli integrita, co když se něco změní, přidá
    - Opět se budeme bavit u vkládání > co chceme ať se stane když v bance omylem zadá jiný příkaz? Asi mi nebude stačit když údaj pouze přepíšů zpět…
    - Už by se nedohledalo…. Můžete zapřemýšlet do příště jak by se mohlo řešit ;)
- Tj. banky – úvěry a účty, hypotéky.
    - Přijde vám dopis na jinou adresu ač u účtu máte změněno, hypotéka o tom neví …
    - Zaměstnanec bude odcházet, tak ať nemá veškerá data (i když jsou firmy, kde politika je opačná, mají přístup ke všemu, platům atd.)
- Ono když se víc se v tabulkách hledá… Mohu se podívat na platby mezi účty kdy nejprve chodí platba z jednoho na druhý a pak je to naopak. ;) napovím podnájmy a muž a žena …
- Jsme zpět u potřeb uživatelů, mohou mít „nelogické“ požadavky na vyhledávání a spojování tabulek.

1.2.9 Transakce

Představte si banku a situaci, kdy jeden bankéř bude odepisovat (strhávat vám platbu) a další bude připisovat výplatu ;) když to bude v jeden okamžik ….
Samozřejmě se bude ***týkat vkládacích příkazu***. Tvorba databázi mezi tyto příkazy zatím nepatří (alespoň v MySQL). V praxi se databáze nemění pod rukama.

- Možná jste si všimly u bank > nelze se připojit probíhá údržba databáze.
    - Čištění, optimalizace, indexace…
- Holt jsou příkazy, které nelze provádět za běhu (ono získat chybou milion na účet by se nemuselo všem líbit)

1.3 Syntaxe CREATE

```
CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] db_name
    [create_specification] ...

create_specification:
    [DEFAULT] CHARACTER SET [=] charset_name
  | [DEFAULT] COLLATE [=] collation_name


SHOW COLLATION              -- JAKÉ JSOU KODOVÁNÍ K DISPOZICI
SHOW CHARACTER SET; -- NEBO JAKÉ JSOU KODOVÁNÍ K DISPOZICI
SHOW CHARACTER SET LIKE 'utf%'
```

1.3.1 Teorie k vytváření databázi. Kodování a řazení

Co se týče češtiny, můžete se nejčastěji setkat se třemi znakovými sadami označovanými (tj. jak budou uloženy české/slovenské znaky):

- latin2 …. (ISO Latin 2),
- cp1250 (Windows 1250)
- utf8 (UTF-8) – funguje i s**lovenština** pro řazení pokud se správně nastaví

Dále je potřeba nastavit i řazení pro češtinu tak máme k dispozici například

- **cp1250_czech_cs** v kódování Windows 1250 řazení,
- **latin2_czech_cs** v kódování ISO 8859-2 řazení
- **utf8_czech_ci** v kódování UTF-8 řazení.
- **utf8_slovak_ci** Pro slovenštinu pak v kódování UTF-8 řazení
- utf8_general_ci Obecné řazení v kódování UTF-8

Pokud není určeno jinak, server MySQL 5.0.x používá ve výchozí konfiguraci znakovou sadu ISO **Latin 1** a řazení **latin1_swedish_ci** (protože domovskou zemí MySQL je Švédsko).

Aby to nebylo tak jednoduché ;)

1.3.2 Klíčové operace

MySQL 5.0.x umožňuje definovat znakovou sadu pro tři klíčové operace.

- jakou znakovou sadu používá klient: character_set_ tedy v jakém kódování přicházejí textová data.
- Proměnnácharacter_set_connectioninformuje server o tom, v jaké znakové sadě požadujete, aby server data zpracovával.
- proměnnácharacter_set_resultsinformuje server o tom, v jaké znakové sadě požadujete, aby server posílal zpracovaná data na výstup.

Pokud tedy například nastane situace, kdy váš klient bude umět pracovat pouze s daty ve Windows 1250, vaše databáze bude umět některé operace (třeba uživatelské procedury) korektně provádět pouze s UTF-8 a váš kontrolní výstup používá ISO Latin 2, můžete následující sadou příkazů instruovat MySQL o svých požadavcích:

1.4 Jak na nastavení

Syntaxe pro zjištění co vaše databáze podporuje

- SHOW CHARACTER SET;
- SHOW CHARACTER SET LIKE 'utf%'

```
SHOW COLLATION;
```

```
+----------+-----------------------------+---------------------+---------+
| Charset  | Description                 | Default collation   | Maxlen |
+----------+-----------------------------+---------------------+---------+
| big5     | Big5 Traditional Chinese    | big5_chinese_ci     |      2 |
| koi8r    | KOI8-R Relcom Russian       | koi8r_general_ci    |      1 |
| latin1   | cp1252 West European        | latin1_swedish_ci   |      1 |
| latin2   | ISO 8859-2 Central European | latin2_general_ci   |      1 |
| swe7     | 7bit Swedish                | swe7_swedish_ci     |      1 |
| ascii    | US ASCII                    | ascii_general_ci    |      1 |
……
| greek    | ISO 8859-7 Greek            | greek_general_ci    |      1 |
| cp1250   | Windows Central European    | cp1250_general_ci   |      1 |
| gbk      | GBK Simplified Chinese      | gbk_chinese_ci      |      2 |
| latin5   | ISO 8859-9 Turkish          | latin5_turkish_ci   |      1 |
+----------+-----------------------------+---------------------+---------+
```

Syntaxe pro zjištení klíčových oblastí

```
show variables like 'char%'

SET character_set_client = cp1250;
SET character_set_connection = utf8;
SET character_set_results = latin2;
```

Pozor nemá nic společného s tím, jakou znakovou sadu či kolaci používá vaše databáze, její tabulky nebo jednotlivé sloupce. Server vše překóduje za běhu podle svých potřeb a nastavení.

Teorií máme za sebou pojďme na databaze tvorbu.

1.5 Začínáme

Nejdříve inspirace a kontrola.

1.5.1 Vypsat databáze, které mám

Syntaxe: Vypíše databáze

```
SHOW DATABASES;
```

1.5.2 Použití databáze

- Php my admin neprovede nutno se prokliknout

Syntaxe: než začnu tvořit

```
USE databáze;
```

1.5.3 Která je aktivní databáze

Zkontroluji, zda opravdu jsem v té databázi, kterou potřebuji

Syntaxe …. Tady klikem (je opravdu aktivní, jsem v té správné?

```
SELECT DATABASE();
```

>> NULL když není vybrána žádná

1.5.4 Nastavení databáze

Syntaxe …. Tady klikem jak je nastaven jazyk a řazení…

```
SELECT @@character_set_database, @@collation_database;
```

- Mít vybranou databázi…

1.5.5 Tvorba databáze

Tak a vytvoříme naší první databázi. Než vytvoříme pár pravidel.

- **Je nás honě tak název si dávejte své UČO**!
- Prosím nemažte si své data – kontrolujte v které databázi jste
- Uvidíme jak bude fungovat ;)

Vytvoří se prázdná databáze, budeme muset v ní vytvořit tabulky, a do tabulek zadat údaje.

Syntaxe: Přístupné databáze

```
create database 777777;  -- UČO!!!
-- nebo
create SCHEMA 777777;
```

co se stane Pokud chci oddělené jméno ?

```
create database `7777 7777`';   -- zpětné lomítko!!!
```

- Alt + 096

Jen upozornění … tabulka neobsahuje nastavení jazyka atd.

Syntaxe: Přístupné databáze pokud již existuje jak ošetřit

```
CREATE DATABASE 777777;
CREATE DATABASE IF NOT EXISTS test;  -- ošetření neskončí chybou
CREATE SCHEMA …
```

1.5.6 Když potřebuji zároveň jazykovou sadu

Syntaxe: Přístupné databáze

```
CREATE DATABASE MojePrvniDatabaze CHARACTER SET utf8 COLLATE
utf8_general_ci; --- texty v textovém souboru

CREATE DATABASE IF NOT EXISTS MojePrvniDatabaze CHARACTER SET utf8 COLLATE
utf8_general_ci;
```

Kodování nastavit **i pro tabulky**, povíme si za chvíli.

1.5.7 Syntaxe ALTER DATABASE / SCHEMA

Co když už mám databázi vytvořenou ať nemusím mazat nebo tvořit novou. Navíc pokud změním v průběhu tak se změní jen nově vytvořené tabulky (budou s novým kodováním).

```
ALTER {DATABASE | SCHEMA} [db_name]
    alter_specification ...
ALTER {DATABASE | SCHEMA} db_name
    UPGRADE DATA DIRECTORY NAME

alter_specification:
    [DEFAULT] CHARACTER SET [=] charset_name
  | [DEFAULT] COLLATE [=] collation_name
```

Syntaxe: **Nastavení jazykové sady** opožděně ne vždy zafunguje

```
ALTER DATABASE <database_name> CHARACTER SET utf8 COLLATE utf8_unicode_ci;
```

>> musí jít o správnou databázi

```
ALTER SCHEMA …
>> někde místo DATABAZE > SCHEMA (někdy obojí)
```

1.6 Přejmenování databáze

- pokud prázdná vytvoříme novou
- pokud obsahuje tabulky vytvoří se nová a „přehodí se tabulky z původní

Syntaxe:

```
CREATE database new_db_name;
** RENAME TABLE db_name.table1 TO new_db_name, db_name.table2 TO
new_db_name;
** DROP database db_name;
```

- ukážeme si až budeme mít tabulky

1.7 Tvorba databáze v PHPmyAdmiin

Ukázka v php adminu…

>> Nová

Vyplnit jméno a „řazení“ z řazení si převezme databáze i kodování

1.8 Nastavení práv pro databázi

Vytváření práv – všichni máte přístup ke všem tabulkám.
Pro správnou by byl potřeba restart a přihlašovat se přes své přihlašovací jména.

Ukážeme si při tvorbě uživatelů. Zde jen poznámka:

Syntaxe:

```
CREATE DATABASE `mydb` CHARACTER SET utf8 COLLATE utf8_general_ci;
GRANT ALL ON `mydb`.* TO `username`@localhost IDENTIFIED BY 'password';
```

1.9 Import / Export

Opět se budeme bavit v samostatné hodině
Exportovat prázdnou databázi nemá smysl ☹

1.10 Mazání databáze

- nelze vrátit zpět

Poslední část.
Mazání databáze. Mnohé systémy neumožňují, člověk musí fyzicky na disk a tam patřičné soubory, složky smazat.

MySQL umožňuje ☹
- vymaže celou databázi se jménem "nazev_databaze", tedy všechny tabulky a data v nich uložená!!

Prosím mažte si jen své databáze. Pozor na jméno.

Syntaxe: Smazání databáze

```
DROP DATABASE 7777;
```