Relační databáze

Úvod přednášek pro distanční výuku

Motivační úloha

Vytvořme si malý domácí informační systém o našich nákupech a spotřebě. Z hromady účtenek, které jsou dnes prodávající povinni vystavit, čerpejme základní data, doplňme je o naše další informace. Budeme očekávat, že náš informační systém odpoví na otázky podobné následujícím:

- Kolik jsme utratili v únoru 2020?
- Kolik jsme utratili v únoru 2020 v Kauflandu?
- Kolik jsme utratili celkem za zeleninu v největším obchodním řetězci?
- O kolik nabobtnal státní rozpočet z daní našich nákupů?

Zatím bez analýzy úlohy a bez bližšího vysvětlení: tušíme, že pro odpovědi na otázky budeme určitě potřebovat alespoň následující data pro jednotlivé položky jednotlivých nákupů:

- Datum, kdy jsme nakoupili.
- Kolik to stálo.
- Popis toho, co to vlastně bylo.
- V jakém obchodě to bylo.
- Jakého druhu bylo to zboží.

Pokud bychom zaznamenali data v tabulkovém procesoru Excel, mohl by záznam z účtenek vypadat třeba následovně:

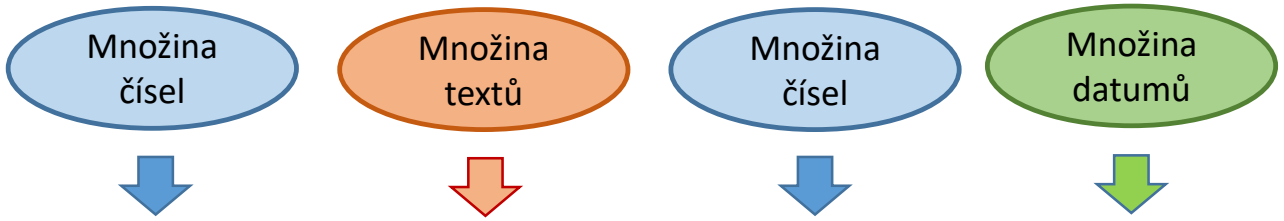
DATUM	KC	CO	MNOZSTVI	JEDNOTKA	DRUH	OBCHOD
1.8.2020	3,00	Housky	2,000	ks	CH	K
1.8.2020	99,90	Mucha Brut	0,750	l	AL	L
1.8.2020	19,80	Minerálky Mattoni	3,000	l	NN	K
1.8.2020	69,90	Kuře grilované	1,000	ks	PO	K
1.8.2020	149,90	Bohemia Brut	0,750	l	AL	L
1.8.2020	6,90	Okurky Salátová	1,000	ks	ZE	K
1.8.2020	17,40	Chléb normální	0,600	kg	CH	T
1.8.2020	11,60	Perník	4,000	ks	CH	T
…	…	…	…	…	…	…

Protože právě program Excel je široce používaný pro zpracování dat, budeme na něj a jeho vlastnosti občas poukazovat. Pro ty, kteří teprve do databázové problematiky pronikají, je však hledání analogií mezi logikou tabulkových procesorů a databázových systémů často (ne)bezpečná cesta ke zmatení.

Data relačních databází

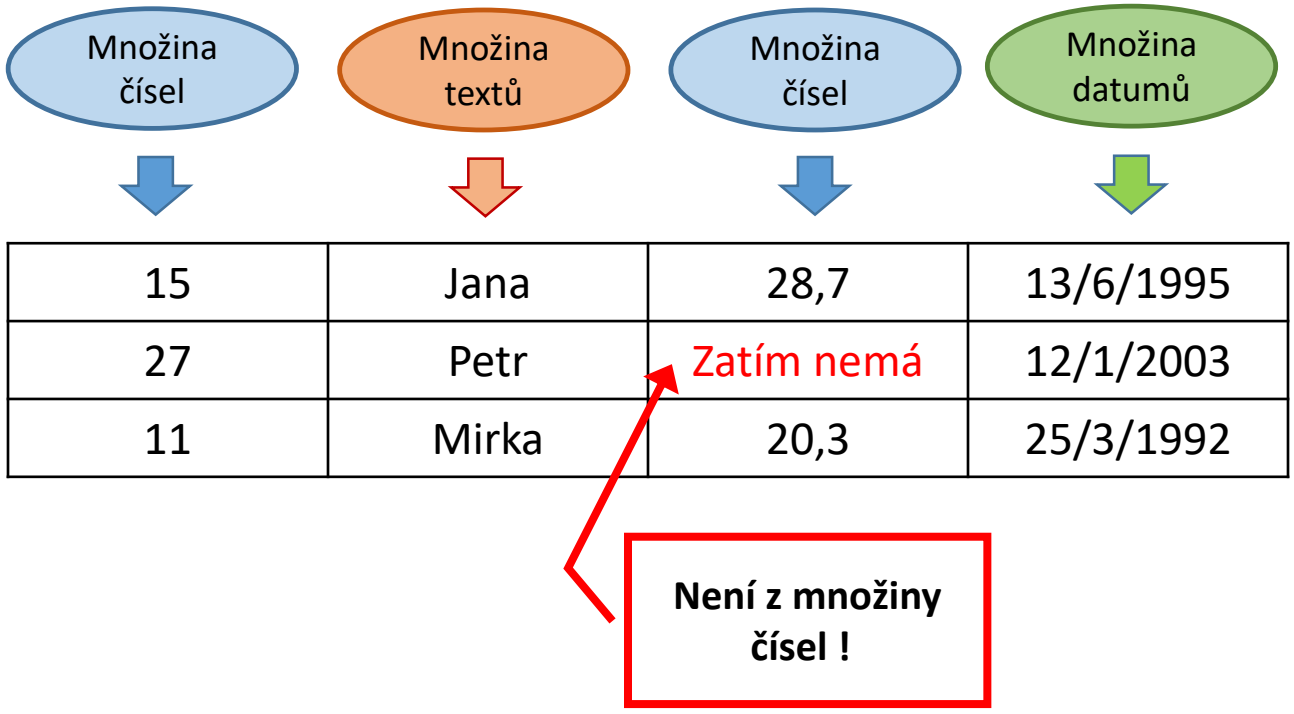
Z pohledu laika pracují relační databáze s daty uspořádanými do obyčejné tabulky. Ta má řádky a sloupce, v databázích jsou sloupce povinně opatřeny nadpisy – ty však nepatří ke zpracovávaným datům.

Relační databáze jsou však založeny na teorii množin. Vychází z pojmu *Kartézský součin* (několika množin) a z pojmu *Relace* jako podmnožiny kartézského součinu. Tabulka jako taková je relací z nějakého kartézského součinu, každý řádek tabulky je prvkem relace. Data jednoho řádku tvoří tedy uspořádanou n-tici, kde n je počet sloupců tabulky. Každý prvek této n-tice je prvkem konkrétní množiny:



15	Jana	28,7	13/6/1995
27	Petr	49,2	12/1/2003
11	Mirka	20,3	25/3/1992

Právě předchozí schéma je tou nejpodstatnější charakteristikou tabulek relačních databází: data každého sloupce pochází ze stejné tzv. **doménové** množiny. V databázích tedy nemůže nastat následující situace:



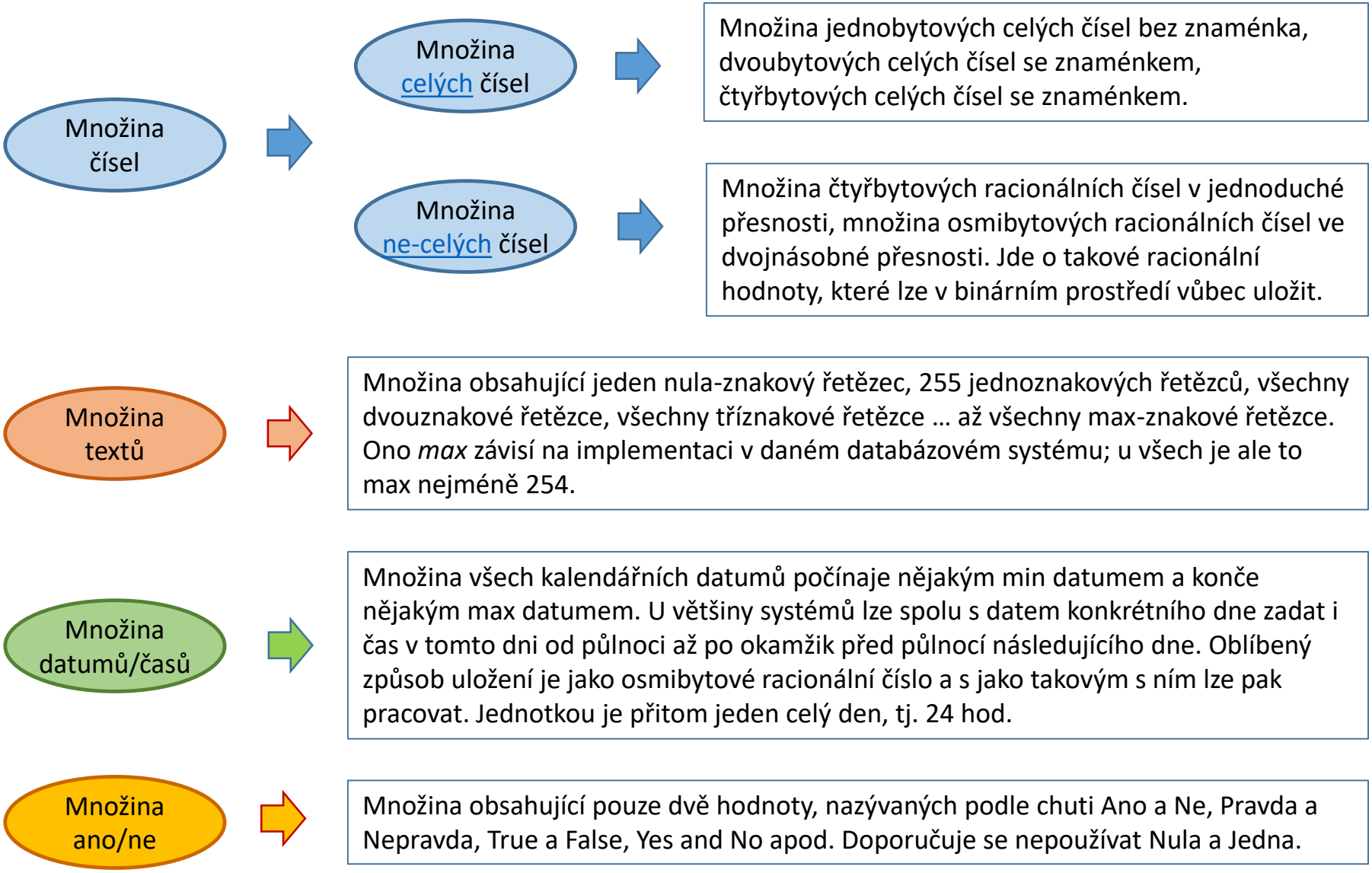
To je jeden z podstatných rozdílů mezi tabulkou Excelu a tabulkou databáze: v databázích je typ dat vázán na celý sloupec, kdežto v Excelu na každou jednotlivou buňku.

Doménové množiny

Doménová množina obsahuje přípustné hodnoty ve sloupci, kterému je přiřazena. V praxi jde vždy o *konečné* (byť třeba „hodně velké“) množiny. Tedy prvky konkrétní doménové množiny mají něco společného, co je do množiny zařazuje. Taková určující vlastnost se s ohledem na použití nazývá **typ dat**. Pro vytváření a používání databázového systému je tedy především nutno tento typ dat přesně specifikovat.

Různé databázové systémy mohou různé typy dat specifikovat různě, byť je dnes zřetelná snaha datové typy unifikovat. Rozdíly, které jsou napříč různými systémy, většinou pramení z různého hardwarového prostředí, ve kterém jsou provozovány.

Uvedeme ty typy dat, které najdeme ve většině systémů bázovaných procesory Intel a jejich klonů. Na případné rozdíly mezi implementacemi upozorníme. Rovnou však upozorňujeme, že v následujícím výčtu byla snaha o průnik typů mezi různými systémy: existuje řada dalších datových typů daných kategorií používaných některými systémy – ale některými naopak ne.


Množina čísel
Množina celých čísel
Množina jednobytových celých čísel bez znaménka, dvoubytových celých čísel se znaménkem, čtyřbytových celých čísel se znaménkem.
Množina ne-celých čísel
Množina čtyřbytových racionálních čísel v jednoduché přesnosti, množina osmibytových racionálních čísel ve dvojnásobné přesnosti. Jde o takové racionální hodnoty, které lze v binárním prostředí vůbec uložit.
Množina textů
Množina obsahující jeden nula-znakový řetězec, 255 jednoznakových řetězců, všechny dvouznakové řetězce, všechny tříznakové řetězce … až všechny max-znakové řetězce. Ono *max* závisí na implementaci v daném databázovém systému; u všech je ale to max nejméně 254.
Množina datumů/časů
Množina všech kalendářních datumů počínaje nějakým min datumem a konče nějakým max datumem. U většiny systémů lze spolu s datem konkrétního dne zadat i čas v tomto dni od půlnoci až po okamžik před půlnocí následujícího dne. Oblíbený způsob uložení je jako osmibytové racionální číslo a s jako takovým s ním lze pak pracovat. Jednotkou je přitom jeden celý den, tj. 24 hod.
Množina ano/ne
Množina obsahující pouze dvě hodnoty, nazývaných podle chuti Ano a Ne, Pravda a Nepravda, True a False, Yes and No apod. Doporučuje se nepoužívat Nula a Jedna.

Množina čísel

Množina celých čísel

Byte – **Byte**: <0, 255>
Integer – **Celé číslo**: <-32 768, +32 767>
Long – **Dlouhé celé číslo**: ± 2 mld.

Množina ne-celých čísel

Single – **Jednoduchá přesnost**, max. 7 cifer
Double – **Dvojnásobná přesnost**, max. 17 cifer

Množina textů

String – **Text**: nutno určit „šířku sloupce“, max. počet znaků údaje

Množina datumů/časů

Date / Time – **Datum a čas**: při zadávání dat lze jen datum, jen čas nebo obojí

Množina ano/ne

Logical – Yes/No **– Ano / Ne**:

Struktura databáze

Z hlediska fyzického uložení svých komponent se databáze liší systém od systému. Co se týče logické struktury, lze v databázích vidět alespoň následující:

- Tabulky dat
- Definice dotazů
- Definice klíčů a vazeb

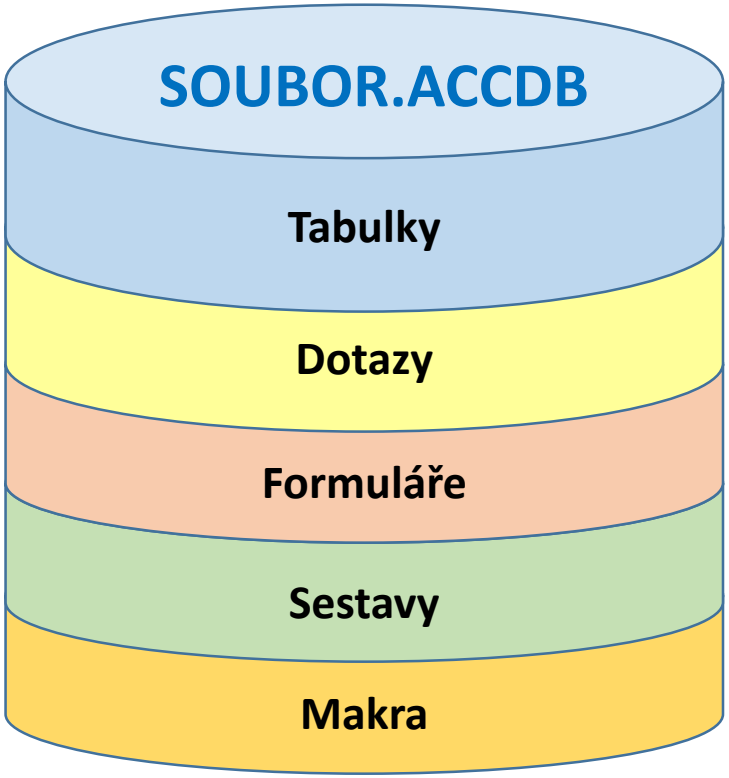
Mnohé databázové systémy nabízí svým návrhářům a zkušenějším uživatelům ještě další komponenty, jako třeba:

- Návrhy formulářů
- Návrhy výstupních sestav
- Definice vlastních (výpočetních) funkcí

Většina databázových systémů obsahuje pro své návrháře tu více, tu méně povedená návrhová prostředí pro jednotlivé komponenty, pro jejich definici resp. změnu.

Struktura databáze Microsoft Jet

Distanční výuka se odehrává v databázovém prostředí Microsoftu, používá konkrétně model nazvaný autory Microsoft Jet. Jeho strukturu lze schematicky přiblížit:



Vlastní data jsou pouze v části Tabulky (včetně popisu struktur tabulek).

V části Makra jsou texty uživatelských funkcí a podprogramů v jazyku VBA.

V ostatních částech jsou *návrhy* příslušných komponent.

Práce s daty, práce se strukturou

Na rozdíl od tabulkových procesorů (Excel) pracují databázové systémy s daty tabulek, které mají v okamžiku zpracování pevně definovanou strukturu – pevně daný počet sloupců, jejich názvy, typ dat v každém sloupci event. upřesnění „šířky“ sloupce. Dále, při zpracování dat jsou přístupná data vždy jen jednoho konkrétního (tzv. *aktuálního*) řádku dat. Který to je, určuje nástroji databázového programu uživatel.

Jakmile se však ukončí režim zpracování dat (jeden režim práce databázového systému), lze databázový systém využívat ve druhém režimu, v režimu práce se strukturou. V něm lze např. přidávat sloupce, rozšiřovat je apod. Databázový systém se přitom snaží zachovat již vložená data. Mnoho změn struktury je však pro data nebezpečných. Zmenším-li šířku textového sloupce, databázový systém bez náhrady „uřízne“ přebývající znaky. Odstraním-li ze struktury sloupec, odstraní se samozřejmě i jeho data.

Proto je třeba provádět změnu struktury velmi uvážlivě; zvláště proto, že se změny v uživatelském návrhovém prostředí provádí velmi jednoduše.

Jako příklad uveďme schematicky náležitosti definice struktury cvičného tématu o nakupování, a to jeho tabulky s daty výdajů:

Jméno sloupce	Datový typ	Velikost pole	Poznámka
DATUM	Datum / Čas		
KČ	Číslo	Jednoduchá přesnost	Max. přesně 99.999,99 Kč
CO	Text	60	Velikost podle potřeby
MNOŽSTVÍ	Číslo	Jednoduchá přesnost	
JEDNOTKA	Text	10	Velikost podle potřeby
DRUH	Text	2	Velikost podle potřeby
OBCHOD	Text	1	Velikost podle potřeby

Následně se tyto texty věnují (až na výjimky) už jen práci s daty v nejrůznějších kontextech. Pro obecnější pojmy budou voleny ukázky dat ze cvičné databáze ÚTRATA – viz motivační úloha v úvodu.

Přístup k datům

K datům přistupuje uživatel vždy pomocí nějakého software. Především to jsou poměrně obecné databázové programy (typu Access) s nabízenými funkcemi, od jejichž počtu a kvality se odvíjí cena. Právě tento typ software budeme mít v pozadí na mysli při výkladu vlastností databází.

Databázové programy by měly uživatelům sedícím u stroje nabízet alespoň

- možnost „ručního“ vkládání dat a jejich pozdější úpravu,
- prohlížení existujících dat v pořadí, jak byla data zadávána,
- prohlížení existujících dat v zadaném pořadí,
- prohlížení dat se zadanou podmínku výběru (= jen některé řádky),
- zobrazovat řádky dat s možností zobrazení hodnot vypočtených z dat řádku,
- možnost seskupovat řádky a zobrazovat hodnoty vypočtených z jednotlivých skupin,
- zobrazovat výsledky dotazů uživatelem (nebo obratnějším kolegou) připravených.

Databázové programy tyto nebo podobné funkce zajišťují jednak v interaktivním režimu prostřednictvím formulářů různého typu, jednak většinou poskytují možnost definovat posloupnost akcí a následně je (třebas opakovaně) provádět v dávkovém režimu.

Filtrování

Pod pojmem *Filtrování* (Filtering, Selection) se rozumí zpracování jen některých řádků dat. Proces filtrování vyžaduje zadání podmínky, kterou je možno vyhodnotit pro každý jednotlivý řádek samostatně. Do zpracování pak vstoupí jen ty řádky, pro které je podmínka splněna.

Způsob zadání podmínky je podmíněn použitým databázovým software a kontextu použití. Jinak uživatel zadá podmínku programu Access, jinak mé vlastní databázové aplikaci. Jinak uživatel zadá podmínku programu Access při prohlížení dat v jeho prostředí datového listu, jinak při použití programu Access jako prostředníka při vydávání databázových dotazů.

Část výsledku filtrování v prostředí datového listu po zadání podmínky uživatelem ve tvaru **KČ>50** může vypadat následovně:

DATUM	KC	CO	MNOZSTVI	JEDNOTKA	DRUH	OBCHOD
DATUM	KC	CO	MNOZSTVI	JEDNOTKA	DRUH	OBCHOD
1.8.2020	99,90	Mucha Brut	0,750	l	AL	L
1.8.2020	69,90	Kuře grilované	1,000	ks	PO	K
1.8.2020	149,90	Bohemia Brut	0,750	l	AL	L
...	...	...	...	...	...	...

Řazení

Pod pojmem *Řazení* (Sorting) se v databázovém dávnověku rozuměl proces, kdy na vstupu byla tabulka a na výstupu tabulka, obsahující tatáž data, ale ve stanoveném pořadí řádků (!). Zdánlivě celkem pochopitelné jsou pak v té souvislosti pojmy řazení *vzestupné* a *sestupné*. Ono stanovené pořadí se od té doby nazývá *kritérium řazení*. V nejjednodušším případě je kritériem řazení jedno datové pole. Např. tabulka VÝDAJE seřazená vzestupně podle kritéria KČ:

DATUM	KC	CO	MNOZSTVI	JEDNOTKA	DRUH	OBCHOD
1.8.2020	3,00	Housky	2,000	ks	CH	K
1.8.2020	6,90	Okurky Salátová	1,000	ks	ZE	K
1.8.2020	11,60	Perník	4,000	ks	CH	T
1.8.2020	17,40	Chléb normální	0,600	kg	CH	T
1.8.2020	19,80	Minerálky Mattoni	3,000	l	NN	K
1.8.2020	69,90	Kuře grilované	1,000	ks	PO	K
1.8.2020	99,90	Mucha Brut	0,750	l	AL	L
1.8.2020	149,90	Bohemia Brut	0,750	l	AL	L
...	...	...	...	...	...	...

Jen na okraj: seřazení hierarchické databáze 6.000 studentů VŠB podle jednoho až pěti kritérií (celkem nezáleželo na počtu kritérií) trvalo v magnetopáskovém systému Tesla-200 dvě až tři hodiny! Proces fyzického řazení podle zvoleného algoritmu spočívá v postupném násobném porovnávání kritérií a fyzického „přehazování“ celých řádků, časová náročnost je pak nasnadě.

Indexové tabulky, indexové klíče

Časem se ukázalo, že požadavek na fyzickou reorganizaci tabulky např. pro potřeby řazení je zbytečně silný. Zvláště v dnešních poměrně rychlých interaktivních systémech uživatel potřebuje ne data **mít** seřazená v určitém pořadí, ale data **vidět** seřazená v určitém pořadí. Mít a vidět, to jsou dva zcela odlišné požadavky splnitelné zcela odlišnými nástroji.

Stejně tak je něco zcela jiného data **mít** seřazená v určitém pořadí, a data **zpracovat** seřazená v určitém pořadí. Při bližším zamyšlení lze tuto disproporci vyřešit stejnými nástroji jako předešlou.

Začněme nejprve nejnázornější situací: Uživatel potřebuje vidět data nějaké tabulky v nějakém, uživatelem stanoveném pořadí - např. VÝDAJE podle ceny KČ. Inu řekne svému databázovému programu: Zobraz mně VÝDAJE seřazené podle KČ. A databázový program mu je předloží ve svém zobrazovacím prostředí (viz také výše):

DATUM	KC	CO	MNOZSTVI	JEDNOTKA	DRUH	OBCHOD
1.8.2020	3,00	Housky	2,000	ks	CH	K
1.8.2020	6,90	Okurky Salátová	1,000	ks	ZE	K
1.8.2020	11,60	Perník	4,000	ks	CH	T
1.8.2020	17,40	Chléb normální	0,600	kg	CH	T
1.8.2020	19,80	Minerálky Mattoni	3,000	l	NN	K
1.8.2020	69,90	Kuře grilované	1,000	ks	PO	K
1.8.2020	99,90	Mucha Brut	0,750	l	AL	L
1.8.2020	149,90	Bohemia Brut	0,750	l	AL	L
...	...	...	...	...	...	...

Pro pochopení lze na to pohlížet takto: Jakmile uživatel poprvé požádá o práci s tabulkou v určitém pořadí, vytvoří databázový systém něco, co lze nazvat *indexovou tabulkou*. Zde např. pro požadavek na práci s tabulkou VÝDAJE v pořadí podle KČ.

Vychází se z toho, že každý záznam (=řádek) tabulky má své fyzické umístění, dané pořadím při zaznamenávání - lidově číslem řádku. Například pro tabulku VÝDAJE (na obrázku nikoliv podle všech cvičných dat)

VÝDAJE

Č.řádku	DATUM	KC	CO	MNOZSTVI	JEDNOTKA	DRUH	OBCHOD
1	01.08.2020	3,00	Housky	2,000	ks	CH	K
2	01.08.2020	99,90	Mucha Brut	0,750	l	AL	L
3	01.08.2020	19,80	Minerálky Mattoni	3,000	l	NN	K
4	01.08.2020	69,90	Kuře grilované	1,000	ks	PO	K
5	01.08.2020	149,90	Bohemia Brut	0,750	l	AL	L
6	01.08.2020	6,90	Okurky Salátová	1,000	ks	ZE	K
7	01.08.2020	17,40	Chléb normální	0,600	kg	CH	T
8	01.08.2020	11,60	Perník	4,000	ks	CH	T
...	...	...	...	...	...	...	...

vytvoří indexovou tabulku (vždy se dvěma sloupci), na kterou lze **velmi zjednodušeně (!!)** pohlížet podle schématu vpravo. První sloupec obsahuje seřazené hodnoty kritéria řazení. Ve druhém je číslo řádku, ve kterém se tato hodnota vyskytuje – tedy jakási vazba na příslušný řádek dat:

Indexová tabulka podle KČ

KC	Č.řádku
3,00	1
6,90	6
11,60	8
17,40	7
19,80	3
69,90	4
99,90	2
149,90	5
...	...

Po vytvoření indexové tabulky (např. na základě uživatelova požadavku na přístup k datům v nějakém pořadí) připojí – předřadí – databázový systém indexovou tabulku před tabulku dat. „Ukazovátko“ na aktuální řádek se pak bude pohybovat nikoliv před datovou, ale před indexovou tabulkou (ty vazby na příslušný řádek dat nejsou pro přehlednost kresleny všechny):

VÝDAJE

KC	Č.řádku
3,00	1
6,90	6
11,60	8
17,40	7
19,80	3
69,90	4
99,90	2
149,90	5
…	…

Č.řádku	DATUM	KC	CO	MNOZSTVI	JEDNOTKA	DRUH	OBCHOD
1	01.08.2020	3,00	Housky	2,000	ks	CH	K
2	01.08.2020	99,90	Mucha Brut	0,750	l	AL	L
3	01.08.2020	19,80	Minerálky Mattoni	3,000	l	NN	K
4	01.08.2020	69,90	Kuře grilované	1,000	ks	PO	K
5	01.08.2020	149,90	Bohemia Brut	0,750	l	AL	L
6	01.08.2020	6,90	Okurky Salátová	1,000	ks	ZE	K
7	01.08.2020	17,40	Chléb normální	0,600	kg	CH	T
8	01.08.2020	11,60	Perník	4,000	ks	CH	T
…	…	…	…	…	…	…	…

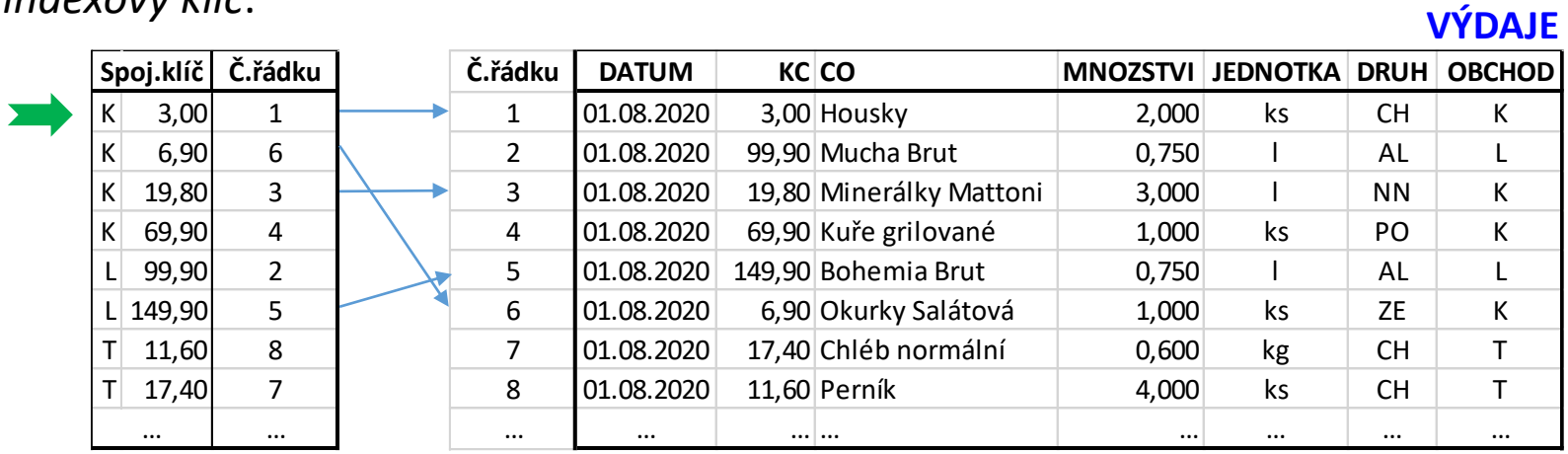
Jako první se zpracuje první řádek podle indexové tabulky, což je (shodou okolností rovněž) první řádek dat. Jako druhý se zpracuje druhý řádek podle indexové tabulky, což je šestý řádek dat – atd.

Pokud uživatel – zůstaňme u příkladu pro účely zobrazení – následně požaduje vidět svá data v pořadí např. podle MNOŽSTVÍ, vytvoří databázový systém analogicky *druhou* indexovou tabulku, v prvním sloupci však budou tentokrát seřazené hodnoty MNOŽSTVÍ. Odpojí indexovou tabulku podle KČ a připojí indexovou tabulku podle MNOŽSTVÍ. „Ukazovátko“ na aktuální řádek se pak bude pohybovat před ní.

Terminologie: Hodnoty v prvním sloupci indexové tabulky se označují pojmem *indexový klíč*. Tedy podle předchozího textu byly vytvořeny dvě indexová tabulky, jedna s indexovým klíčem KČ, jedna s indexovým klíčem MNOŽSTVÍ.

Uživatel si však může vymyslet pohled na data (resp. jejich zpracování) v pořadí podle více kritérií, např. podle OBCHODU a KČ současně. Nejprve se seřadí řádky podle prvního kritéria (zde podle OBCHODU) a jednotlivé skupiny řádků se stejným prvním kritériem (se stejným OBCHODEM) se samostatně seřadí podle druhého kritéria (zde podle KČ).

I když v takovém případě už existují indexové tabulky podle KČ a případně by existovala i podle OBCHODU, neexistuje způsob, jak obě indexové tabulky „zkombinovat“. Databázový systém vytvoří novou indexovou tabulku, opět se dvěma sloupci, kde však hodnoty prvního sloupce (= indexový klíč) budou tvořeny hodnotami OBCHODU spojenými s hodnotami KČ (téhož řádku dat), tzv. *spojený indexový klíč*:



Aktualizace indexových tabulek

Vytvořené indexové tabulky zůstávají součástí databáze, pokud je uživatel nástroji databázového programu nesmaže. Dále může těmito nástroji požádat o „odpojení“ indexové tabulky; tím se vrátí pořadí zpracování řádků k původnímu pořadí (ve kterém byla data zadávána).

Tato koncepce by měla mít jeden nepříjemný důsledek: uživatel může kdykoliv měnit kterákoliv svá data v kterékoliv tabulce. Změní-li data v nějakém sloupci, podle kterého byla vytvořena indexová tabulka, už většinou nebude odpovídat pořadí v indexové tabulce pořadí ve vlastních datech.

Databázové systémy tuto situaci řeší tak, že kdykoliv dojde ke změně hodnoty v nějakém sloupci, automaticky (bez nějaké další činnosti uživatele) přeindexuje všechny indexová tabulky, ve kterých se tato hodnota vyskytovala jako klíč nebo jeho součást.

Práce s indexovými tabulkami je maximálně optimalizována a tedy velmi rychlá. Většinou k přeindexování dojde okamžitě po změně hodnoty uživatelem, ten to časově ani nepostřehne.

Vlastnosti indexových tabulek

Vytvořené indexové tabulky mohou mít z hlediska hodnot indexových klíčů různé vlastnosti. Nejčastěji jsou brány v potaz dvě vlastnosti klíče:

- Jedinečnost klíče (UNIQUE Key): Tuto vlastnost má takový klíč, ve kterém se žádná hodnota nevyskytuje dvakrát nebo vícekrát. Klíč shora uváděného příkladu indexové tabulky podle OBCHODU určitě nemá tuto vlastnost (několikrát jsme nakupovali ve stejném obchodě).

  Pokud už v tabulce dat existují zadané hodnoty, pak stanovit pro klíč vlastnost jedinečnosti nelze na základě toho, zda data *teď* této vlastnosti vyhovují, ale jestli i v budoucnu jí *budou* vyhovovat. Např. nestanovím v tabulce VÝDAJE sloupec KČ jako jedinečný klíč, protože bych si pak nemohl koupit dvě piva, každé za 21 Kč.

- Neprázdnost klíče (The key is NOT NULL): Tuto vlastnost má takový klíč, který v žádné své části není prázdný - NULL se používá ve smyslu nikoliv nuly, ale jako indikace prázdné, nezadané hodnoty. Česky tato vlastnost zní lépe jako „Všude vyplněný“.

Primární klíč - indexová tabulka

Pokud má nějaký indexový klíč obě výše zmíněné vlastnosti:

- je jedinečný,
- je neprázdný,

pak může být prohlášen za tzv. *primární* klíč a indexová tabulka, jejímž je klíčem, za *primární* indexovou tabulku dané tabulky dat. Tím se současně tato indexová tabulka připojí - předřadí tabulce dat, jejíž zpracování pak bude touto indexovou tabulkou řízeno. Primárním klíčem může být např. prohlášen sloupec KÓD tabulky OBCHODY.

V daném okamžiku může být pro jednu tabulku dat definována (připojená) jediná primární indexová tabulka. Jako všechny indexové tabulky lze tabulka, která má být prohlášena za primární, vytvořit a smazat. V některých systémech však nejde samostatně připojit a odpojit - odpojit jedině tak, že ji smažu.

Je-li pro nějakou tabulku dat definován primární klíč, pak databázový systém reaguje chybou (a databázový program by to měl se svým uživatelem vyřešit), jsou-li porušena pravidla pro primární klíč. V případě primárního klíče KÓD tabulky OBCHODY to nastane v případě, kdy uživatel buď smaže třebas kód K jako Kaufland, nebo přidá další řádek s kódem K jako Kika - teda ten už vlastně není, že ☺.

Vazby mezi tabulkami - Relace

V úvodu je uvedena motivační úloha s otázkou: Kolik našich peněz odvedli prodejci jako DPH z našich nákupů? Výpočet DPH, to je jednoduchá trojčlenka - známe-li ovšem sazbu daně, která se však liší podle druhu zboží. V celém předchozím textu jsme však na tuto otázku zapomněli, proto se vrátíme na začátek a do struktury tabulky přidáme ještě sloupec se sazbou daně (níže už je i ukázka použití této daně):

DATUM	KC	CO	MNOZSTVI	JEDNOTKA	DRUH	PROC_DANE	DPH	OBCHOD
1.8.2020	3,00	Housky	2,000	ks	CH	15	0,39	K
1.8.2020	99,90	Mucha Brut	0,750	l	AL	21	17,34	L
1.8.2020	19,80	Minerálky Mattoni	3,000	l	NN	15	2,58	K
1.8.2020	69,90	Kuře grilované	1,000	ks	PO	15	9,12	K
1.8.2020	149,90	Bohemia Brut	0,750	l	AL	21	26,02	L
1.8.2020	6,90	Okurky Salátová	1,000	ks	ZE	15	0,90	K
1.8.2020	17,40	Chléb normální	0,600	kg	CH	15	2,27	T
1.8.2020	11,60	Perník	4,000	ks	CH	15	1,51	T
...	...	...	...	...	...	...	...	...

Tím jsme si ovšem zadělali na velký problém. Budou volby, po nich už nebude daň z chlastu 21%, ale 33%. Při zpětné simulaci stávajících výdajů s novou daní by bylo třeba nahradit hodnotu 21 hodnotou 33, ale jen u alkoholu. Při množství vypitém průměrným Čechem za rok by to byla vpravdě neúnosná dřina.

Sazba daně použitá způsobem z předchozího textu je klasickou ukázkou tzv. *redundantního* údaje: údaje, který se se stejnou hodnotou a stejným významem opakuje v tabulce dat několikrát. Příkladů je bezpočet. Např. příjmení zaměstnance v personální agendě se nemůže vyskytnout mnohokrát v jedné tabulce. Nováková je pěkné děvče, taky ji oblouznil Jožo Černý - a Nováková už není Nováková ale Černá. Kde hledat a nahradit všechny výskyty příjmení Nováková? Nedej bože, že ve firmě mám dvě Novákové, nemůžu přeci udělat z Joži bigamistu!

V databázových aplikacích je zásadou mít jeden takový údaj jedinkrát na jediném místě. Pakliže je třeba mít sazbu DPH několikrát v jedné tabulce TX, nelze mít v této tabulce přímo hodnotu sazby - ta musí být tedy v jiné tabulce TY jedinkrát, logicky v jediném řádku oné jiné tabulky. V tabulce TX pak nebude hodnota sazby, ale něco, z čeho lze odvodit ten řádek tabulky TY, ve kterém se hodnota sazby nachází.

Jinak řečeno: ve zmíněných tabulkách TX a TY musí být něco společného, co se během provozování informačního systému nebude měnit. Jde o obecně známý kód položky: osobní číslo, UserName apod.

V ukázkové databázi se sazba DPH 21% vztahuje ke druhu zboží **AL**kohol; to je zaznamenáno ve sloupci DRUH. Abychom se tolik nenamáhali při psaní do tabulky VÝDAJE, od počátku tam píšeme dvoupísmenovou zkratku druhu. Pokud tedy budeme sazby DPH různých druhů zboží uvádět v další tabulce (třeba DRUHY), můžeme právě tuto zkratku použít jako vazební prvek mezi dvěma tabulkami:

VÝDAJE

Vazba, Relace (jed.č.)

DATUM	KC	CO	MNOZSTVI	JEDNOTKA	DRUH	OBCHOD
1.8.2020	3,00	Housky	2,000	ks	CH	K
1.8.2020	99,90	Mucha Brut	0,750	l	AL	L
1.8.2020	19,80	Minerálky Mattoni	3,000	l	NN	K
1.8.2020	69,90	Kuře grilované	1,000	ks	PO	K
1.8.2020	149,90	Bohemia Brut	0,750	l	AL	L
1.8.2020	6,90	Okurky Salátová	1,000	ks	ZE	K
1.8.2020	17,40	Chléb normální	0,600	kg	CH	T
1.8.2020	11,60	Perník	4,000	ks	CH	T
...	...	...	...	...	...	...

DRUHY

KOD	NAZEV	PROC_DANE
AL	Alkoholické nápoje	21
CU	Cukr, cukrářské výrobky	15
CH	Chléb, pečivo	15
KA	Káva, čaj	15
...	...	...

Naznačená vazba je v databázi jedna mnoha, které je možno definovat. Lze třeba definovat analogickou relaci z tabulky VÝDAJE do tabulky OBCHODY přes pole OBCHOD (už budou relace - množ. číslo - dvě) . Jinou relací může být vazba opačná: z tabulky DRUHY do tabulky VÝDAJE. Pak při procházení tabulky DRUHY máme postupně přístupný každý VÝDAJ tohoto DRUHU. Blíže však PROPOJENÍ v kapitole DOTAZY.

Při procházení tabulky, ze které vychází vazba (zde VÝDAJE) se ke každému jejímu řádku *logicky* připojí ten řádek tabulky, do které směřuje vazba (zde DRUHY) a který má stejný KÓD jako je hodnota v poli DRUH:

VÝDAJE

DATUM	KC	CO	MNOZSTVI	JEDNOTKA	DRUH	OBCHOD
1.8.2020	3,00	Housky	2,000	ks	CH	K
1.8.2020	99,90	Mucha Brut	0,750	l	AL	L
1.8.2020	19,80	Minerálky Mattoni	3,000	l	NN	K
1.8.2020	69,90	Kuře grilované	1,000	ks	PO	K
1.8.2020	149,90	Bohemia Brut	0,750	l	AL	L
1.8.2020	6,90	Okurky Salátová	1,000	ks	ZE	K
1.8.2020	17,40	Chléb normální	0,600	kg	CH	T
1.8.2020	11,60	Perník	4,000	ks	CH	T
…	…	…	…	…	…	…

DRUHY

KOD	NAZEV	PROC_DANE
AL	Alkoholické nápoje	21
CU	Cukr, cukrářské výrobky	15
CH	Chléb, pečivo	15
KA	Káva, čaj	15
…	…	…

Bude-li se zpracovávat první řádek tabulky VÝDAJE, budou v ten okamžik dostupné nejen všechny hodnoty v tomto řádku, ale také všechny hodnoty v odkazovaném řádku tabulky DRUHY. Tedy nejen dvě housky z Kauflandu, ale i jejich 15% daň. Stejně tak budu-li zpracovávat Muchu z Lidlu (datově; fyzicky jsem to už dávno udělal), budu mít k disposici jeho 21% daň.

Vazby = relace definované mezi tabulkami se stávají součástí databáze (uživatel má samozřejmě možnost kteroukoliv kdykoliv upravit nebo smazat, má-li k tomu oprávnění).

Vlastnosti relací

Vytvářené relaci lze - při splnění jistých předpokladů - definovat požadavky, jejichž uplatnění zajišťuje databázový systém. Jedny z nejsilnějších požadavků lze definovat tehdy, směřuje-li vazba do toho sloupce cílové tabulky, na základě kterého byla vytvořena (a připojena) primární indexová tabulka. Hodnoty tohoto sloupce tedy vytváří primární indexový klíč a mají dvě jeho vlastnosti: jedinečnost a vyplněnost.

V příkladu z předchozí stránky to nastane v případě, je-li pro tabulku DRUHY definováno datové pole KÓD jako primární klíč. To jistě lze (pokud ho v nějakém řádku nezapomenu vyplnit). Přeci v tabulce DRUHY nezadám dva řádky se stejným KÓDEM! Jak bych pak v tabulce VÝDAJE poznal, jestli je to ten první druh výdaje nebo ten druhý?

Je-li tedy cílové pole určeno jako primární klíč cílové tabulky, lze po databázovém systému požadovat požadavky na

- kontrolu referenční integrity,
- aktualizaci v kaskádě,
- vypuštění v kaskádě.

Pokus o vysvětlení těchto záhadných pojmů následuje. Bude učiněn na zmíněném příkladu vazby z tabulky VÝDAJE do tabulky DRUHY.

Kontrola referenční integrity

Databázový systém v tomto případě kontroluje, zda to, co je zadáváno (nebo bude zadáváno) do sloupce DRUH v tabulce VÝDAJE, je obsaženo někde ve sloupci KÓD tabulky DRUHY. Pokud tedy v nějakém řádku výdajů zkusím zapsat jako druh XY, pak databázový systém vyvolá chybu, protože v tabulce DRUHY žádný kód XY není.

Na chybu by měl umět reagovat databázový program, za jehož chodu k chybě došlo. Např. program Access v prostředí datového listu přeruší uživateli práci s daty, zobrazí mu upřesnění chyby, a dá mu na výběr dvě možnosti: buď tam program Access vrátí hodnotu, která tam byla předtím, nebo nechá uživateli jako další pokus zadat jinou hodnotu. Snad se uživatel strefí do něčeho přípustného.

Aktualizace v kaskádě

Situace: potřebuju změnit druh výdaje PO=Pomeranče na OR=Orange, protože PO chci nově použít jako druh výdaje PO=Pojistné (mnozí divoši na PO=Pokuty). To je složité! Nemůžu v tabulce DRUHY změnit PO na OR, protože pomeranče ve výdajích by neměly kam ukazovat, a nemůžu ve výdajích začít měnit PO na OR, protože bych porušoval referenční integritu.

Právě tuto situaci řeší aktualizace v kaskádě: stačí, abych v tabulce druhy změnil kód PO na OR, a databázový systém automaticky změní v tabulce VÝDAJE všechny druhy PO na OR a na nic - na rozdíl ode mě - nezapomene.

Vypuštění v kaskádě

Situace: mám tentokrát vytvořenu další relaci, z VÝDAJŮ do OBCHODŮ. Kdysi v dávné minulosti jsem nadšeně nakupoval v obchodě C jako Carrefour, který sem hned po revoluci jako jeden z prvních vtrhnul - a jako jeden z prvních odešel, protože zjistil, že tady mu pšenka nekvete tak jako jinde. Proč tedy schovávat desítky let staré výdaje z tohoto obchodu?

Řeší to jednoduše databázový systém (srov. předchozí Aktualizaci v kaskádě). Požadavkem na vypuštění v kaskádě smaže systém všechny řádky tabulky VÝDAJE mající jako kód obchodu ten, který jsem já právě smazal (přesněji jeho řádek odstranil) v tabulce OBCHODY. A to ihned a navždy.

Jednoduché - a nebezpečné. Stačí, abych měl v tabulce OBCHODY označený nějaký řádek a omylem se opřel do klávesy Delete. A spolehlivý databázový systém ihned zajistí, že mám celou databázi v jednom neslušném místě. A nemám-li funkční kopii, jsem tam taky.