Databáze standardu SQL, díl 10.

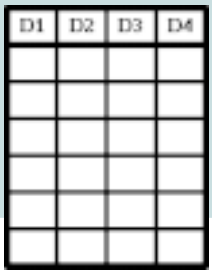
Agregace v SQL

Naším dnešním úkolem bude dokončit povídání o užitečnosti hodnoty NULL a abychom neztráceli čas, směle se pustíme do agregačních funkcí.

Jistá firma se rozhodla nabídnout své služby těm firmám, které ještě nejsou připojeny na internet. Pro jednoduchost použila následující příkaz:

```
SELECT ICO,NAZEV,ADRESA FROM FIRMA WHERE EMAIL IS NULL;
```

Tak vznikla obrovská tabulka potenciálních zákazníků. Protože nebylo dost peněz na poštovné, bylo rozhodnuto firmy omezit jen na ty, které vykazují roční zisk alespoň jeden



Obr. 1. Tabulka s nerozlišenými řádky.

milion korun. Tak vznikl lepší výběr firem příkazem:

```
SELECT ICO,NAZEV,ADRESA FROM FIRMA WHERE EMAIL IS NULL AND ZISK>=1000000;
```

Bohužel takových firem nebylo příliš mnoho, protože některé firmy tají svůj zisk, aby je nikdo neunavoval. Chceme-li je zahrnout do marketingové akce, použijeme příkaz:

```
SELECT ICO,NAZEV,ADRESA FROM FIRMA WHERE EMAIL IS NULL
        AND ( ZISK>=1000000 OR ZISK IS NULL);
```

Pokud by nevhodně vzrostl počet firem ve výběru, můžeme být ještě rafinovanější:

```
SELECT ICO,NAZEV,ADRESA FROM FIRMA WHERE EMAIL IS NULL
        AND ( ZISK>=1000000 OR ZISK IS NULL AND ZAMESTNANCI>25);
```

Nyní je na vás slovně interpretovat předchozí příklad. Hned potom se vžijme do situace zdravotní sestry, která chce pozvat do poradny všechny dosud neočkované děti a pomocí následujícího příkazu zjistit:

```
SELECT * FROM KOJENEC WHERE NOT OCKOVAN;
```

Příkaz by bylo vhodné trochu vylepšit s použitím testování nedostatku informace:

```
SELECT * FROM KOJENEC WHERE NOT OCKOVAN OR OCKOVAN IS NULL;
```

Pokud v kartotéce uchováváme i údaje o dětech mezitím vyřazených z evidence, což je někdy výhodné, zamezíme nepříjemnostem drobným vylepšením příkazu na tvar:

```
SELECT * FROM KOJENEC WHERE ( NOT OCKOVAN OR OCKOVAN IS NULL )
        AND VYRAZEN_DNE IS NULL;
```

V jednom odtučňovacím sanatoriu se rozhodli vypracovat abecední seznam lidí s nadváhou větší než 30 kg pro podchycení zájemců o absolutní čtyřtýdenní dietu. Naivně použili příkaz:

```
SELECT PRIJMENI,JMENO,NAROZEN FROM NAIVNI_HOST
WHERE 100+VAHA-VYSKA>30 ORDER BY PRIJMENI,JMENO;
```

Někteří pacienti totiž tajili své kritické parametry. Tak vznikl velmi rafinovaný příkaz:

```
SELECT PRIJMENI,JMENO,NAROZEN FROM NAIVNI_HOST
WHERE 100+VAHA-VYSKA>30 OR VAHA IS NULL OR VYSKA IS NULL
ORDER BY PRIJMENI,JMENO;
```

Podmínka výběru šla napsat i jiným způsobem:

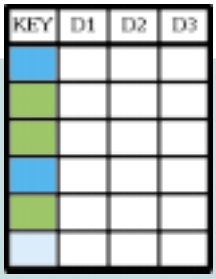
```
SELECT PRIJMENI,JMENO,NAROZEN FROM NAIVNI_HOST
WHERE 100+VAHA-VYSKA>30 OR 100+VAHA-VYSKA IS NULL
ORDER BY PRIJMENI,JMENO;
```

Obr 2. Výsledek agregace bez rozlišování řádků.

Elegantní by potom bylo seznam setřídit podle nadváhy. Na to potřebujeme dva příkazy SQL:

```
UPDATE NAIVNI_HOST SET NADVAHA=100+VAHA-VYSKA;

SELECT NADVAHA,PRIJMENI,JMENO, NAROZEN FROM NAIVNI_HOST
WHERE NADVAHA>30 OR NADVAHA IS NULL
ORDER BY NADVAHA DESC,PRIJMENI, JMENO;
```

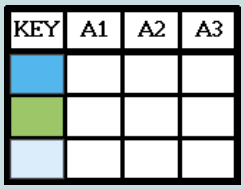
Pro další seznámení s možnostmi výrazů s NULL slouží malá soutěž o nejlepšího kocoura, organizovaná tak, že vyhraje kocour, který se nejvíc blíží takto definovanému ideálu: je hodný, čistotný, není líný a je přitom tajemný. Za každou vlastnost počítáme 10 bodů. Pokud je vlastnost utajena, pak za ni kocour obdrží jenom 1 bod. Za celkovou tajemnost se dává navíc 13 bodů. Poslední příklad slouží k demonstraci restrikce v rámci příkazu UPDATE:



Obr. 3. Tabulka s rozlišenými řádky.

```
UPDATE KOCOUR SET BODY=0;

UPDATE KOCOUR SET BODY=BODY+10 WHERE HODNY;

UPDATE KOCOUR SET BODY=BODY+10 WHERE CISTOTNY;

UPDATE KOCOUR SET BODY=BODY+10 WHERE NOT LINY;

UPDATE KOCOUR SET BODY=BODY+10 WHERE TAJEMNY;

UPDATE KOCOUR SET BODY=BODY+1 WHERE HODNY IS NULL;

UPDATE KOCOUR SET BODY=BODY+1 WHERE CISTOTNY IS NULL;

UPDATE KOCOUR SET BODY=BODY+1 WHERE LINY IS NULL;

UPDATE KOCOUR SET BODY=BODY+1 WHERE TAJEMNY IS NULL;

UPDATE KOCOUR SET BODY=BODY+13 WHERE HODNY IS NULL
```

Obr. 4. Výsledek agregace podle klíče.



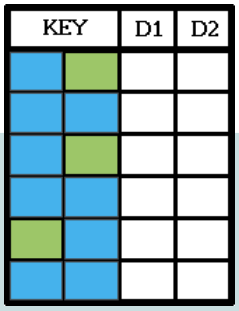

```
AND CISTOTNY IS NULL AND LINY IS
NULL AND TAJEMNY IS NULL;

SELECT BODY, JMENO, MAJITEL FROM
KOCOUR
ORDER BY BODY DESC, JMENO;
```

Jako návnada k četbě dalších dílů seriálu je uveden příkaz obsahující agregační funkci a zahnízděný příkaz. Jen tak se nám podaří vypsat nejlepší kocoury, kteří se dělí o první místo v soutěži:

```
SELECT JMENO, MAJITEL FROM KO-
COUR
WHERE BODY= ( SELECT MAX(BODY)
FROM KOCOUR )
ORDER BY JMENO,MAJITEL;
```

Agregace

Ten, kdo nechce v rámci SQL dotazu vidět všechno, může pomocí projekce a restrikce zmenšit počet sloupců a řádků zobrazené tabulky. Tak je možné se zaměřit na podstatné detaily problému. Někdy nás nezajímá ani rozsáhlý celek, ani jeho detail, ale spíše celkové hodnocení. Průměrný, maximální a minimální plat zaměstnance jsou někdy důležitější informace než konkrétní znalost Quidova platu nebo přehled všech platů. Postupu, který nahradí skupinu řádků tabulky jedním řádkem,



Obr. 5. Tabulka se složeným klíčem.

říkáme shlukování neboli AGREGACE. Předpisu pro výpočet hodnoty ve shluku říkáme AGREGAČNÍ FUNKCE. Proces agregace můžeme jednoduše znázornit graficky. Nejjednodušší situace nastává, když nerozlišujeme jednotlivé řádky výchozí tabulky a ze všech řádků vytvoříme jeden agregát. Na obrázku 1 je znázorněna výchozí tabulka a na obrázku 2 je výsledek agregace jako tabulka obsahující právě jeden řádek. Ten pak obsahuje jednotlivé hodnoty agregačních funkcí. Kromě již uvedené minimální, maximální a průměrné hodnoty jsou v SQL k dispozici ještě dvě agregační funkce pro součet a pro počet všech agregovaných hodnot.

Ne vždy však sčítáme hrušky se švestkami. Prakticky se snažíme vytvořit v rámci jedné tabulky více agregátů a v nich pak samostatně vyčíslit hodnoty agregačních funkcí. Na obrázku 3 jsou barevně odlišeny různé hodnoty v jednom klíčovém sloupci. Z toho plyne, že nejde o unikátní klíč pro přímý přístup, ale o klíč k agregaci řádků. Pokud budeme chtít sledovat prodej zboží v závislosti na druhu, budeme agregáty vytvářet podle klíčového sloupce CISLO_ZBOZI. Zajímá-li nás celkový obrat po měsících, zvolíme klíčový sloupec MESIC. Pokud chceme zjistit naše největší a nejmenší odběratele, zvolíme pro tutéž tabulku klíčový sloupec ICO. Stejná barva ve sloupci KEY znamená stejnou hodnotu například čísla zboží. Na obrázku 4 vidíme výsledek skupinové agregace podle klíče. Vznikly pouze tři řádky představující tři dílčí výsledky agregace podle stejného čísla zboží.

Obr. 6. Agregace se složeným klíčem.

Pokud nás bude zajímat celkový přehled o odběratelích v jednotlivých měsících, budeme muset kombinovat sloupce ICO a MESIC do složeného klíče pro agregaci. Při sledování sezonních prodejů zboží má smysl klíč složený ze sloupců CISLO_ZBOZI a MESIC. Pro pochopení struktury trhu se hodí použít kombinaci sloupců CISLO_ZBOZI a ICO. Na obrázku 5 je symbolicky znázorněna tabulka se složeným klíčem ICO a MESIC. Z barevného značení je vidět, že ve dvou měsících nakupovaly pouze dvě firmy a přitom nastaly tři různé kombinace určující řádky k agregaci. Na obrázku 6 je výsledek agregace podle složeného klíče.

Víme tedy, co je to agregace a proč je užitečná. Zbývá jí realizovat v rámci SQL příkazu SELECT. V tabulce 1 jsou vysvětlena nová klíčová slova a funkce jazyka SQL. Za symbolem funkce jsou vždy uvedeny závorky, uvnitř kterých je uveden výraz, tedy většinou jenom název sloupce.

Agregace bez klíče

Pokud výchozí tabulka obsahuje úplná data a nehodláme rozlišovat jednotlivé řádky, je

SQL	význam
AVG()	průměr platných hodnot
COUNT(*)	počet řádků
COUNT()	počet platných hodnot
DISTINCT	pouze odlišné hodnoty
GROUP BY	seskupit podle klíče
HAVING	mající
MAX()	maximum platných hodnot
MIN()	minimum platných hodnot
SUM()	součet platných hodnot

Tabulka 1. Klíčová slova SQL – agregace.

použití agregace jednoduchou záležitostí. V tabulce 2 jsou uvedena výchozí data.

K porozumění významu agregačních funkcí postačí použít je na obsah sloupce B pomocí příkazu

```
SELECT COUNT(B), SUM(B), AVG(B),
MIN(B), MAX(B) FROM FULL_TAB;
```

Tak vznikne tabulka 3 obsahující právě jeden řádek s hodnotami agregačních funkcí. Po řadě obsahuje hodnotu počtu platných hodnot B, součtu, aritmetického průměru, minimální a maximální hodnotu.

Při agregaci není nutné se omezovat jen na jeden sloupec. Agregační funkce je možné použít na jakýkoli výraz odkazující se na libovolný počet sloupců původní tabulky. Pro

A	B	C
1	5	0
2	1	1
3	6	10
4	3	7

Tabulka 2. Obsah tabulky FULL_TAB.

ilustraci pomůže SQL příkaz, který navíc přejmenuje výrazy v záhlaví výsledné tabulky 4:

```
SELECT COUNT(A*C) P, SUM(A*C) Q,
AVG(A*C) R, MIN(A*C) S,MAX(A*C) T
    FROM FULL_TAB;
```

Velmi podstatné je chování agregačních funkcí vůči nedefinovaným hodnotám NULL. Zde platí pravidlo, že nedefinované hodnoty se nezapočítávají do počtu, součtu, průměru, minima či maxima. V porovnání s předchozím výkladem, kde NULL v aritmetických výrazech způsobí nedefinovanou hodnotu výsledku, jde vlastně o další výjimku. Poslední vý-

COUNT(B)	SUM(B)	AVG(B)	MIN(B)	MAX(B)
4	15	3,75	1	6

Tabulka 3. Agregace obsahu sloupce B.

P	Q	R	S	T
4	60	15	0	30

Tabulka 4. Agregace a výrazy.

P	Q	R
NULL	10	3
NULL	NULL	NULL
13	2	NULL
13	3	NULL

Tabulka 5. Obsah tabulky NULL_TAB.

jimkou je pak agregační funkce COUNT(*), která zjistí pouhý počet řádků bez ohledu na nedefinovanost jejich obsahu. V tabulce 5 je ukázka neúplných dat. Po provedení příkazu

```
SELECT COUNT(*),COUNT( P ),COUNT( Q IS NULL ),COUNT( P/R ),COUNT( DISTINCT P )
FROM NULL_TAB;
```

obdržíme tabulku 6.

Jen ve dvou řádcích ze čtyř je definován výraz P. Výraz Q IS NULL je definován vždy a má hodnotu YES, nebo NO. Naproti tomu výraz P/R není definován v ani jednom řádku. Konečně výraz P, přestože má ve dvou řádcích definovanou hodnotu, má hodnotu jenom jednu, a ta se rovná 13. V tabulce 7 jsou uvedeny příklady chování zbylých agregačních funkcí na nedefinovaných datech. Taková tabulka vznikne příkazem:

```
SELECT SUM(P),AVG(Q), MIN( P+7*Q ), MAX(R),MAX( P-R ) FROM NULL_TAB;
```

Povšimněte si průměru ze tří hodnot Q, jejichž součet je 15. Výraz P+7*Q nabývá dvou konkrétních hodnot 27 a 30, takže je jasné, která je menší. Maximum z jediné hodnoty R je přirozeně tato hodnota. Konečně není-li v žádném řádku definován výraz P-R, logicky se neví nic ani o něm, ani o jeho maximální hodnotě. Všimněte si, že při agregaci bez klíče je možné zobrazovat pouze hodnoty agregačních funkcí. Ostatní hodnoty jsou víceznačně definovány a nemají smysl.

Agregace s jednoduchým klíčem

Vraťme se k obrázkům 3 a 4. Chceme-li použít jeden sloupec tabulky jako klíč ke shlukování řádků do agregátů, stačí uvést klíčové slovo GROUP BY a za ním název agregačního sloupce. Bývá dobrým zvykem tento sloupec uvést v seznamu zobrazovaných hodnot na prvním místě, a pak se teprve uvádějí hodnoty agregačních funkcí. Pokud nemáme rádi chaos, musíme ještě podle tohoto sloupce setřídit řádky s využitím ORDER BY na konci SQL příkazu. Představme si příkaz:

```
SELECT CO, SUM(PLACENO) CELKEM
FROM NAKUP
GROUP BY CO
ORDER BY CO;
```

Pokud ho použijeme na tabulku 8, obdržíme tabulku 9.

CO	DEN	KOLIK	JEDNOTKA	PLACENO
ANANAS	PONDELI	1,5	kg	45
KURE	PONDELI	0,6	kg	42
MLEKO	UTERY	2	l	30
KURE	UTERY	0,5	kg	30
ANANAS	PATEK	2	kg	60
MLEKO	PATEK	3	l	48
VEJCE	PATEK	10	ks	30

Tabulka 8. Nákup starého mládence.

Jako malou ukázku chybného příkazu, za který nám SQL server vynadá, uvádím následující dotaz:

```
SELECT CO, SUM(PLACENO) CELKEM
FROM NAKUP;
```

Chybí zde GROUP BY, a proto jde o agregaci bez klíče. Hodnota SUM(PLACENO) je definována jako celková hodnota všech nákupů. Hodnota CO však nabývá čtyř různých hodnot, které máme vtěsnat do jednoho pole,

CO	CELKEM
ANANAS	105
KURE	72
MLEKO	78
VEJCE	30

Tabulka 9. Za co se utrácí.

a to není dovoleno. Následující dva příkazy SQL jsou naopak užitečné:

```
SELECT SUM(PLACENO) CELKEM
FROM NAKUP;
```

COUNT(*)	COUNT(P)	COUNT (Q IS NULL)	COUNT (P/R)	COUNT (DISTINCT P)
4	2	4	0	1

Tabulka 6. Možnost agregační funkce COUNT().

SUM(P)	AVG(Q)	MIN(P+7*Q)	MAX(R)	MAX(P-R)
26	5	27	3	NULL

Tabulka 7. Agregační funkce a NULL.

```
SELECT SUM(PLACENO) BEZ_ANANASU FROM NAKUP
WHERE NOT CO=”ANANAS”;
```

CELKEM
285

Tabulka 10. Celková útrata.

BEZ_ANANASU
180

Tabulka 11. Bez ananasových hodů.

První z nich určí celkovou hodnotu všech nákupů rovnou 285 Kč a nazve ji CELKEM. Druhý příkaz odpoví na otázku, kolik by starý mládenec utratil, kdyby nemlsal ananas. V tabulkách 10 a 11 jsou výsledky uvedených dotazů.

Platí jednoduché pravidlo pro zápis dotazu s agregací:

```
SELECT popis projekce FROM tabulka
WHERE podmínka restrikce
GROUP BY klíč agregace
ORDER BY klíč třídění;
```

Jde vlastně o povinný popis projekce a datového zdroje, pak následuje nepovinný popis restrikce, nepovinný klíč k agregaci a nepovinné setřídění podle obecně jiného klíče. Pro pochopení dotazu s agregací je třeba si uvědomit pořadí kroků, které provádí SQL server po obdržení dotazu. Nejprve je proveden výběr tabulky, pak následuje restrikce, která většinou výrazně zmenší počet řádků. Následná agregace se zabývá pouze vybranými řádky po restrikci a podle klíče nebo bez něj sestaví agregáty s využitím agregačních funkcí. Ty jsou setříděny, a teprve potom dojde na projekci jednotlivých sloupců. Tomu odpovídá schéma:

TABULKA => RESTRIKCE => AGREGACE => TŘÍDĚNÍ => PROJEKCE

V SQL je rozpor mezi pořadím v syntaxi a pořadím provádění jednotlivých kroků. To naštěstí mate jen uživatele, ale nemate SQL server, který zkontroluje syntaxi příkazu, a pak zvolí optimální pořadí kroků. Server zejména při velkém množství dat vyhodnocuje dotazy pomalu. Proto často slyšíme diskuse o optimalizaci dotazů. Platí jednoduché optimalizační pravidlo, že všechna pole uvedená v podmínce restrikce, v klíči agregace a v klíči třídění by měla mít svůj index a že výraz popisující restrikci by měl být jednoduchý.

Jaromír Kukal