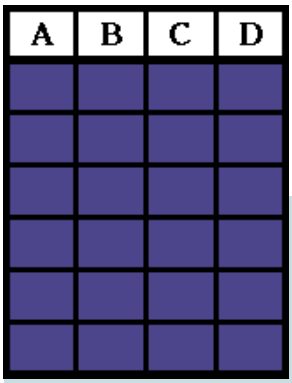
Databáze standardu SQL, díl 8.

Restrikce v SQL

V návaznosti na předcházející díly se dnes konečně budeme zabývat omezováním počtu řádků v jazyce SQL. Tato operace se už od dob Coddových nazývá restrikcí.

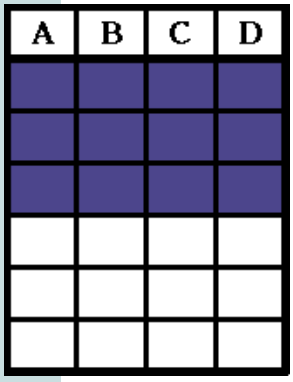
Na obrázku 1 vidíme symbolickou tabulku se čtyřmi sloupci, kde všechny řádky jsou předmětem zkoumání a jsou označeny modrou barvou. Takto jsme doposud postupovali, neboť nás zajímala především projekce hodnot výrazů uvedených v tabulce.



Obr. 1. Bez restrikce je vidět vše.

Při čtení textu předchozího dílu vás asi napadlo, že pokud data třídíme, může to být kvůli tomu, že nás zajímají pouze řádky, ve kterých je dosaženo nejnižšího nebo nejvyššího hodnocení. Pak je možné v souladu s obrázkem 2 zobrazovat jenom první tři položky.

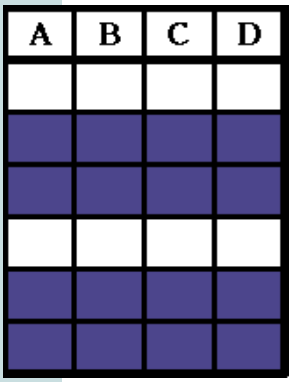
Obr. 2. Tři vítězné řádky.



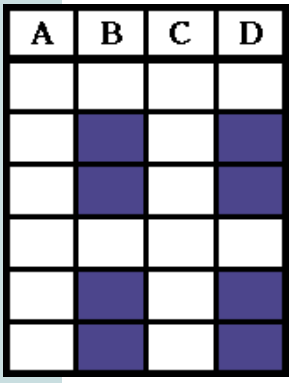
V obecném případě, jak je znázorněno na obrázku 3, nám jde o zobrazování těch položek, respektive řádků tabulky, které mají určitý stěžejní význam, tedy nikoliv jen kvůli jejich dobrému pořadí.

Uvažujme typický případ, kdy původní tabulka obsahuje velké množství informací ve velkém množství řádků a nás upřímně nezajímají nepodstatné informace. O to více si přejeme v úsporné formě vidět podstatné informace. Na obrázku 3 jsou jako podstatné informace chápány informace ve druhém, třetím, pátém a šestém řádku. Vybrat takové řádky na přeskáčku je snadné. Budeme muset formulovat nějakou vhodnou logickou podmínku výběru, která bude dána logickým výrazem. Ty řádky, ve kterých bude mít logický výraz hodnotu YES, budou vybrány. Pokud bude hodnota výrazu NO nebo NULL, příslušný řádek nebude vybrán. Takto definovaná restrikce jako omezování počtu řádků ve výsledné výstupní tabulce je velmi užitečná. Navíc je možné restrikci řádků kombinovat s projekcí sloupců v jednom SQL příkazu. Na obrázku 4 vidíme v symbolické formě kombinaci projekce, kde nás zajímají pouze sloupce B a D

Obr. 3. Obecná restrikce.



Obr. 4. Kombinace projekce a restrikce.



SQL	význam
_	jeden znak
%	skupina znaků
IN	je uvnitř
IS	je
LIKE	vypadá jako
NOT	ne
PERCENT	procent
TOP	horních
WHERE	kde platí

Tabulka 1. Klíčová slova SQL – Restrikce.

s restrikcí na čtyři řádky, které splňují konkrétní podmínku.

Potom jsme schopni se kombinací projekce a restrikce zaměřit jen na ty informace, které doopravdy požadujeme. Při této příležitosti znovu upozorňuji, že málokdy a málokdo chce vidět úplně vše, protože vidět vše v celé řadě praktických případů znamená zcela sám sebe dezorientovat v daném problému. Rozumné vytržení informací z kontextu nezaškodí. Přistupme nyní k výkladu základních principů restrikce v jazyce SQL.

V tabulce 1 vidíme klíčová slova jazyka SQL pro realizaci restrikce. Kromě nich jsou zde i speciální znaky jako _ a % , které se vyskytují v maskách pro zkoumání podobnosti textů. Zkusme si představit, že se nám podařilo příkazem

CREATE TABLE CHAOS (CISLO INTEGER, NAZEV VARCHAR(10));

CISLO	NAZEV
13	SROUBOVAK
7	ROHLIK
31	NAPAD
10	ZATACKA
3	ORION

Tabulka 2. Velký chaos.

vytvořit tabulku s názvem CHAOS o dvou sloupcích. Sloupec CISLO představuje číslo položky a sloupec NAZEV je její pojmenování. Taková tabulka může obsahovat nejrůznější položky v chaotickém uspořádání. Zároveň zde nejsou žádná doménová či entitní integritní omezení. V takové tabulce byl pak několika příkazy INSERT INTO způsoben požadovaný CHAOS. Ten nám poslouží pro pochopení základních principů formulace příkazů pro restrikci v SQL. V tabulce 2 vidíme přímý výpis obsahu tabulky CHAOS po naplnění daty. Postačil příkaz

SELECT * FROM CHAOS;

V přímém výpisu se zatím vyskytují oba dva sloupce a všechny řádky bez možnosti si vybrat. Pokusme se nyní z tabulky CHAOS vybrat ty první tři řádky, které mají nejmenší číslo položky. Patrně při tom budeme kombinovat třídění položek podle sloupce CISLO a zároveň vybereme první tři položky ze setříděného seznamu. To učiníme s využitím klí-

čového slova TOP následovaného číslem 3 tak, jak popisuje příkaz

SELECT TOP 3 * FROM CHAOS ORDER BY CISLO;

CISLO	NAZEV
3	ORION
7	ROHLIK
10	ZATACKA

Tabulka 3. Nejmenší tři.

Výsledek vidíme v tabulce 3 a je plně v souladu s naším původně slovně formulovaným zadáním. Omezovat v setříděných seznamech počet zobrazených položek můžeme též s využitím představy o procentu zobrazených položek. Mějme na paměti, že 100 % rozsahu znamená zobrazit všechny řádky, zatímco jakékoli jiné nenulové menší číslo znamená příslušnou alikvotní část celku. Pokud nás zajímají pouze dvě první položky, můžeme to chápat jako 40 % celkového rozsahu pětiřádkové tabulky. Můžeme formulovat následující příkaz pro výběr prvních 40 % položek z hlediska jejich abecedního uspořádání:

SELECT TOP 40 PERCENT * FROM CHAOS ORDER BY NAZEV;

CISLO	NAZEV
31	NAPAD
3	ORION

Tabulka 4. 40 % podle abecedy.

Výsledek je v tabulce 4. Taková jednoduchá restrikce patrně neuspokojí veškeré naše požadavky na vybírání vhodných řádků. Budeme muset zavést vhodnější formu restrikce, kterou budeme vybírat jednotlivé položky a která nebude založena na třídění a postupu od počátku tabulky. Dostáváme se k dalšímu klíčovému slovu jazyka SQL. Slovo WHERE odděluje projekční část příkazu SELECT od části restriktivní. Za klíčové slovo WHERE vždy píšeme logický výraz. Pokud je výraz roven hodnotě YES, pak je příslušný řádek zobrazen. V ostatních případech zobrazen není. Když chceme z tabulky CHAOS vybrat všechny položky, jejichž evidenční číslo je větší než 10, napíšeme příkaz

SELECT * FROM CHAOS WHERE CISLO>10;

CISLO	NAZEV
13	SROUBOVAK
31	NAPAD

Tabulka 5. Čísla větší než deset.

V tabulce 5 skutečně vidíme položky 13 a 31 a jejich názvy. Pokud bychom se rozhodli přesněji hledat položky podle jejich čísla v tabulce CHAOS, pak v podmínce za WHERE použijeme logický výraz posuzující rovnost hodnoty CISLO a zadané konstanty. Chceme-li najít přesně položku s číslem 10, užijeme příkaz

SELECT * FROM CHAOS WHERE CISLO=10;

CISLO	NAZEV
10	ZATACKA

Tabulka 6. Vím přesně, co chci najít.

V tabulce 6 je vidět, že dané podmínce vyhovuje právě jeden řádek. Pokud jsme takto přísní při použití restrikce, může se stát, že konkrétní hodnota se v tabulce nevyskytuje. Taková neexistující hodnota by mohla být hledána příkazem

SELECT * FROM CHAOS WHERE CISLO=5;

Tím bohužel vznikne zcela prázdná tabulka obsahující pouze záhlaví položek a neobsahující žádné konkrétní vnitřní údaje tak, jak je uvedeno v tabulce 7.

CISLO	NAZEV

Tabulka 7. Ale mám smůlu.

Pro konstrukci výrazu na restrikci můžeme použít operátor IN, jehož použití neznamená nic jiného než testování, zda nějaká hodnota výrazu je prvkem dané množiny hodnot. Nejčastěji píšeme před klíčové slovo IN název sloupce, ve kterém hledáme, a za klíčové slovo IN seznam hodnot oddělených čárkou a uzavřených do obyčejných kulatých závorek. Jsou ještě další komplikovanější možnosti, které budou předmětem příštích dílů seriálu. Hledáme-li zatoulané položky s čísly 11, 3 a 18, pak příslušný příkaz zní:

SELECT * FROM CHAOS WHERE CISLO IN (11,3,18);

Máme částečnou smůlu, neboť položky s čísly 11 a 18 se nevyskytují v původní tabulce CHAOS. To nepovede k žádné havárii, pouze k výběru zbylé položky s číslem 3. Máme tak možnost pomocí klíčového slova IN zkoumat širším způsobem, zda se některá z uvedených hodnot vyskytuje v daném řádku a sloupci tabulky. Výsledná tabulka obsahující jeden řádek je uvedena pod číslem 8.

CISLO	NAZEV
3	ORION

Tabulka 8. Výlov podle množiny.

Pro úplnost je uvedena pracnější možnost docílení téhož efektu bez použití množiny

SELECT * FROM CHAOS WHERE CISLO=11 OR CISLO=3 OR CISLO=18;

Velmi časté je testování, zda se nějaká hodnota vyskytuje v zadaném intervalu hodnot. Na to slouží klíčová slova BETWEEN a AND. Před klíčovým slovem BETWEEN uvádíme nejčastěji název příslušného sloupce, ve kterém je hledáno, za klíčových slovem BETWEEN uvádíme hodnotu dolní meze a za klíčovým slovem AND hodnotu horní meze. Pro ilustraci poslouží příkaz jazyka SQL, který vyhledá všechny položky s čísly od 5 do 10 včetně a zní:

SELECT * FROM CHAOS WHERE CISLO BETWEEN 5 AND 10;

CISLO	NAZEV
7	ROHLIK
10	ZATACKA

Tabulka 9. Použití intervalu.

V tabulce 9 vidíme položky s čísly 7 a 10, které prošly restriktivním příkazem. Téhož efektu bylo možné docílit s využitím tradiční logiky:

SELECT * FROM CHAOS WHERE CISLO>=5 AND CISLO<=10;

Při porovnávání hodnot není nutné se omezovat pouze na hodnoty číselné. Je možné porovnávat hodnoty textových řetězců, hodnoty typu datum či logické hodnoty podle charakteru řešené konkrétní úlohy. Pouze práce s texty přináší nové možnosti konstrukce restriktivních výrazů. Jde o neúplné vyhledávání v textech. Hledáme-li v chaosu naší tabulky položku s názvem ROHLIK, pak nezbude nic jiného než slovo ROHLIK chápat jako textovou konstantu a požadovat rovnost s konstantou pomocí příkazu

SELECT * FROM CHAOS WHERE NAZEV="ROHLIK";

CISLO	NAZEV
7	ROHLIK

Tabulka 10. Hledám podle názvu.

V tabulce 10 potom vidíme požadovanou položku číslo 7. Takové přímé prohledávání nemusí být vždy užitečné. Velmi často hledáme nejrůznější fragmenty textu jen kvůli tomu, že nemáme přesnou informaci o hledaném textu. Pro ilustraci možností takovéhoto vyhledávání je uvedeno několik dalších příkladů. Při neúplném vyhledávání musíme místo rovnítka mezi názvem sloupce, ve kterém je hledáno, a příslušným vyhledávaným textem vložit klíčové slovo LIKE. Chceme tím říci, že nezáleží na přesné shodě textu. Pouze trváme na shodě s polotovarem textu, který umožňuje víceznačnost. Za klíčovým slovem LIKE musí být uvedena textová konstanta v uvozovkách nazývaná též maskou, která obsahuje vyhledávaný fragment textu a řídicí znaky. Pokud v textu nejsou použity řídicí znaky, je příkaz

LIKE formálně shodný s rovnítkem mezi texty. Předchozí příkaz, který vygeneroval tabulku 10, lze napsat též jako

```
SELECT * FROM CHAOS WHERE NAZEV
LIKE ”ROHLIK”;
```

Existují dva speciální znaky umožňující neúplné vyhledávání. První z nich je znak %, který je divokou kartou ve vyhledávacím řetězci. Představuje totiž blíže neurčený počet neznámých znaků. Místo tohoto znaku si můžeme představit N libovolných znaků, kde N může být rovno i nule. Pokusme se vžít do role sklerotika, který přesně neví, jak se jmenují položky v tabulce CHAOS a pouze tuší koncové písmeno K. Před ním může být blíže neurčené množství jiných písmen. Příkaz SQL potom vypadá následovně:

```
SELECT * FROM CHAOS WHERE NAZEV
LIKE ”%K”;
```

CISLO	NAZEV
13	SROUBOVAK
7	ROHLIK

Tabulka 11. Hledám koncové K.

Dojde k vygenerování tabulky 11, ve které jsou uvedeny jen dvě položky končící písmenem K. Pokud bychom místo této úlohy chtěli řešit něco komplikovanějšího, můžeme hledat všechny názvy, ve kterých se někde vyskytuje písmeno O. Masku pro vyhledávání musíme napsat tak, že začíná i končí znakem %. To znamená, že hledaný text může začít blíže neurčeným počtem znaků, potom následuje písmeno O a konečně blíže neurčený počet dalších znaků. Říkáme tak, že před O a za O může být cokoli, tedy i nic. Příkaz hledající všechny názvy s O někde uvnitř je

```
SELECT * FROM CHAOS WHERE NAZEV
LIKE ” %O%”;
```

CISLO	NAZEV
13	SROUBOVAK
7	ROHLIK
3	ORION

Tabulka 12. Hledám O kdekoli.

V tabulce 12 pak vidíme vybrané tři položky, o kterých není pochyb, že obsahují uvnitř slov písmeno O. Máte-li rádi astronomii, pak možná znáte studený sopečný měsíc IO. Který z následujících tří příkazů SQL ho najde, nebo najde alespoň něco jiného?

```
SELECT * FROM CHAOS WHERE NAZEV
LIKE ” IO”;
SELECT * FROM CHAOS WHERE NAZEV
LIKE ” %IO%”;
SELECT * FROM CHAOS WHERE NAZEV
LIKE ” %IO”;
```

Dalším speciálním znakem, který je užitečný při formulaci masky, je znak _, zvaný podtržítko. Není tak velkorysý jako znak %. Řídicí znak podtržítko totiž zastupuje právě jeden znak ve zkoumaném textovém řetězci. Pokud bychom se vrátili k hledání K na konci slova a omylem použili podtržítko místo %, dostaneme příkaz

```
SELECT * FROM CHAOS WHERE NAZEV
LIKE ”_K”;
```

Žádné slovo v tabulce CHAOS neobsahuje pouze dvě písmena, natož aby končilo písmenem K. Proto uvidíme pouze prázdnou tabulku 7. Lidé, kteří rádi luští křížovky, občas hledají slovo, které je na šest písmen a začíná písmenem R. Pokud bychom chtěli takové slovo vybrat z naší chaotické tabulky, stačí napsat příkaz

```
SELECT * FROM CHAOS WHERE NAZEV
LIKE ” R_ _ _ _ _”;
```

a ihned se nám objeví tabulka 10 a v ní jedna jediná položka zvaná ROHLIK. Zkusme ještě domyslet, zda by v některých situacích stálo za to kombinovat jak znak %, tak i podtržítko. Pokud pomocník křížovkáře ví, že druhé písmenko slova je R a poslední písmenko je N, nemusí ani vědět, jak je takové slovo dlouhé. Stačí napsat

```
SELECT * FROM CHAOS WHERE NAZEV
LIKE ” _R%N”;
```

a ihned je vybrána položka ORION tak, jak je uvedeno v tabulce 8. Operátory IN a LIKE se mohou kombinovat s klíčovým slovem NOT pro obrácení jejich významu. Má smysl hovořit o operátorech NOT LIKE a NOT IN, které znamenají „nevypadá jako“ a „není prvkem množiny“. Nesnášíme-li písmenko Y pro jeho tvrdost a chceme vypsat všechny názvy, které ho neobsahují, napíšeme příkaz

```
SELECT * FROM CHAOS WHERE NAZEV
NOT LIKE ” %Y%”;
```

místo logického a také korektního řešení

```
SELECT * FROM CHAOS WHERE NOT
(NAZEV LIKE ” %Y%” );
```

Výsledkem je shodou okolností celá tabulka 2, kde ani jedna položka není vynechána. Pokud jsme po autonehodě alergičtí na některá konkrétní slova, můžeme zformulovat příkaz, který vypíše všechny položky kromě položek, jejichž název je přesně roven specifikovaným slovům

```
SELECT * FROM CHAOS
WHERE NAZEV NOT IN (”SROUBOVAK”,”ROHLIK”,”ZATACKA”);
```

Výsledek je uveden v tabulce 4. Uvedené příklady restrikce byly doposud modelové a vhodným způsobem nás seznámily s možnostmi konstrukce restrikcí. Nyní bude užitečné dokumentovat možnosti restrikce na praktických příkladech.

Pokud bychom ze souboru lidí chtěli vypsat podle abecedy dívky starší 18 let, použijeme příkaz

```
SELECT * FROM LIDI
WHERE ZENA AND VEK >18
ORDER BY PRIJMENI, JMENO;
```

Tento příkaz není zcela uvážený a možná, že příkaz uvedený v následujícím řádku by trochu lépe splnil účel:

```
SELECT * FROM LIDI
WHERE ZENA AND NOT VDANA AND
VEK BETWEEN 19 AND 30
ORDER BY PRIJMENI, JMENO, CPO
DESC;
```

Představte si položpionážní úlohu, kdy se nám podařilo na displeji mobilního telefonu přečíst telefonní číslo anonymního (jak už to bývá) vyděrače. Kdopak nám to vlastně volá a kam poslat ochranku? Hodil by se SQL příkaz

```
SELECT JMENO, PRIJMENI, MISTNOST,
PATRO, CP, ULICE, MESTO FROM PRIS-
TROJ
WHERE TELEFON=”3141592171717”
ORDER BY PRIJMENI, JMENO;
```

Nyní jistě chápete, proč se taková databáze běžně neprodává. Zvýšila by se tím účinnost a pravděpodobnost přímé odplaty. Dále je vám, doufám, jasné, že tabulka PRISTROJ je v 5NF, a tak není možné psát více telefonních čísel do jednoho řádku. Je to tak dobře, protože nám přece jde o přesnou lokalizaci přístroje k zásahu.

V případě, že by nás zajímalo, co nejvíce informací o horních deseti tisících, stačí napsat restrikci

```
SELECT TOP 10000 * FROM LIDI ORDER
BY PRACHY DESC;
```

Při takto slabé restrikci se nadře SQL server a vy to po něm stěží přečtete až do konce. Pokud budete vdávat dceru, bude se hodit rozumnější kombinace projekce a restrikce, aby si mohla kvalifikovaně vybrat a nepokazit si zrak u monitoru:

```
SELECT TOP 10 JMENO, PRIJMENI,
VEK,OCI FROM LIDI
WHERE SVOBODNY AND (NOT DEBIL
OR VEK > 70)
AND PRACHY>10000000
ORDER BY PRACHY DESC;
```

Jaromír Kukal