

Závěrečný sprint

Takřka po dvou letech zkoumání možností SQL databází se dostáváme až do samotného finále.

Než se pustíme do dokončení projektu OZNAMKA, rád bych popsal obecné možnosti řízení výpočtu uvnitř procedur. Nejdůležitější je pojem bloku. BLOK je skupina příkazů sevřených mezi BEGIN a END. Uvnitř procedury je vždy jeden blok tvořící její tělo. Bloky se mohou též zahnížďovat a vytvářet tak struktury. Posloupnost příkazů uzavřená do bloku se totiž nejen v SQL chová jako jeden příkaz. Toho využíváme jak při větvení, tak při cyklech. Příkaz větvení má dva obecné tvary:

IF logický výraz
THEN příkaz provedený nejvýše jednou

IF logický výraz
THEN příkaz provedený při splnění podmínky
ELSE opačný příkaz

Logické výrazy a příkazy už známe dávno a bloky usnadní konverzi posloupnosti příkazů na jeden příkaz. Pro cyklus s testováním podmínky před započítáním práce se používá schéma:

WHILE logický výraz
DO příkaz opakovaný několikrát

Pro cyklus přes všechny řádky tabulky určené příkazem SELECT se používá schéma:

FOR select příkaz
DO příkaz aplikovaný na řádek selectu

Pro ošetření chyb použijeme schéma:

WHEN ANY
DO příkaz číhající na chybu

Pro předčasný východ z procedury použijeme příkaz EXIT.
Použití cyklu, bloku a ošetření výjimek je uvedeno v následujících dvou procedu-

rách. Procedura GAUSS řeší Gaussův školácký problém sečtení čísel od 1 do N. Procedura PRUSVIH řeší stejný problém, ale hledá nejvyšší možné číslo, pro které se výpočet ještě nezhroutí. Všimněte si práce s bloky a lokálními proměnnými:

```
CREATE PROCEDURE GAUSS (N INTEGER) RETURNS (S INTEGER)
AS
DECLARE VARIABLE K INTEGER;
BEGIN
:K=0;
:S=0;
WHILE :K<=:N
DO BEGIN
:S=:S+:K;
:K=:K+1;
END
END
```

```
CREATE PROCEDURE PRUSVIH RETURNS (K INTEGER,S INTEGER)
AS
DECLARE VARIABLE KNEW INTEGER;
DECLARE VARIABLE SNEW INTEGER;
BEGIN
:K=0;
:S=0;
WHILE YES
DO BEGIN
:SNEW=:S+:K;
WHEN ANY
DO EXIT;
:KNEW=:K+1;
:S=:SNEW;
:K=:KNEW;
END
END
```

Konec projektu OZNAMKA

Pod vlivem předchozího výkladu už snadno vytvoříme potřebné uložené procedury. Začneme s tabulkou CLOVEK, pro její údržbu je třeba šest procedur.

Procedura NOVYCLOVEK umožňuje přidat nového člověka a přitom hlídat nenulovost rodného čísla, jména a příjmení spolu s unikátností rodného čísla:

```
CREATE PROCEDURE NOVYCLOVEK(RCX VARCHAR(10),JX VARCHAR(30),PX VARCHAR(30))
AS
BEGIN
IF NOT EXISTS(SELECT RC FROM CLOVEK WHERE RC=:RCX) AND :JX IS NOT NULL
AND :PX IS NOT NULL AND :RCX IS NOT NULL
THEN INSERT INTO CLOVEK(RC,JMENO,PRIJMENI,KOEUD,KOEOB)
VALUES (:RCX,:JX,:PX,1.0,1.0);
END
```

Proceduře NOVYCLOVEK zbývá k dokonalosti už jen logická kontrola rodného čísla :RCX, kterou ponechávám čtenáři k nedatabázovým úvahám. Člověka můžeme zrušit, pokud ještě neudával, respektive nebyl udán tak, jak je uvedeno v proceduře ZRUSCLOVEK:

```
CREATE PROCEDURE ZRUSCLOVEK(RCX VARCHAR(10))
AS
BEGIN
IF NOT EXISTS(SELECT RCUD FROM UDANI WHERE RCUD=:RCX OR RCOB=:RCX)
THEN DELETE FROM CLOVEK WHERE RC=:RCX;
END
```

Změna jména nebo příjmení je proti tomu maličkost, jak vidíme v procedurách ZMENJMENO a ZMENPRIJMENI:

```
CREATE PROCEDURE ZMENJMENO (RCX VARCHAR(10),JX VARCHAR(30))
AS
BEGIN
IF :JX IS NOT NULL
THEN UPDATE CLOVEK SET JMENO=:JX WHERE RC=:RCX;
```

END

```
CREATE PROCEDURE ZMENPRIJME-
NI(RCX VARCHAR(10),PX VAR-
CHAR(30))
AS
BEGIN
IF :PX IS NOT NULL
THEN UPDATE CLOVEK SET PRIJME-
NI=:PX WHERE RC=:RCX;
END
```

Některé oběti mají prominentní postave-
ní a za jejich udání náleží příspěvek ur-
čený koeficientem KOEX větším než jed-
na nebo rovným jedné. Podobně
prominentní udavač bere za své služby
více. Procedury JAKOUBET a JAKOUDA-
VAC se hodí pro aktualizaci příplatků:

```
CREATE PROCEDURE JAKOUBET(RCX
VARCHAR(10),KOEX DECIMAL(10,3))
AS
BEGIN
IF :KOEX IS NOT NULL AND :KOEX
>=1.0
THEN UPDATE CLOVEK SET KOE-
OB=:KOEX WHERE RC=:RCX;
```

END

```
CREATE PROCEDURE JAKOUDA-
VAC(RCX VARCHAR(10),KOEX DECI-
MAL(10,3))
AS
BEGIN
IF :KOEX IS NOT NULL AND
:KOEX>=1.0
THEN UPDATE CLOVEK SET KOE-
UD=:KOEX WHERE RC=:RCX;
END
```

Nyní se budeme zabývat údržbou tabul-
ky CIN, na kterou stačí použít čtyři pro-
cedury. Nejzajímavější z nich je procedu-
ra NOVYCIN, která sama určuje hodnotu
nového primárního klíče CIC a kontroluje
unikátnost názvu činu. Je to praktická
ukázka nepotřebnosti příkazů TRIGGER
při důsledném postupu aktualizace:

```
CREATE PROCEDURE NOVYCIN (NX
VARCHAR(30))
AS
DECLARE VARIABLE CIX INTEGER;
BEGIN
```

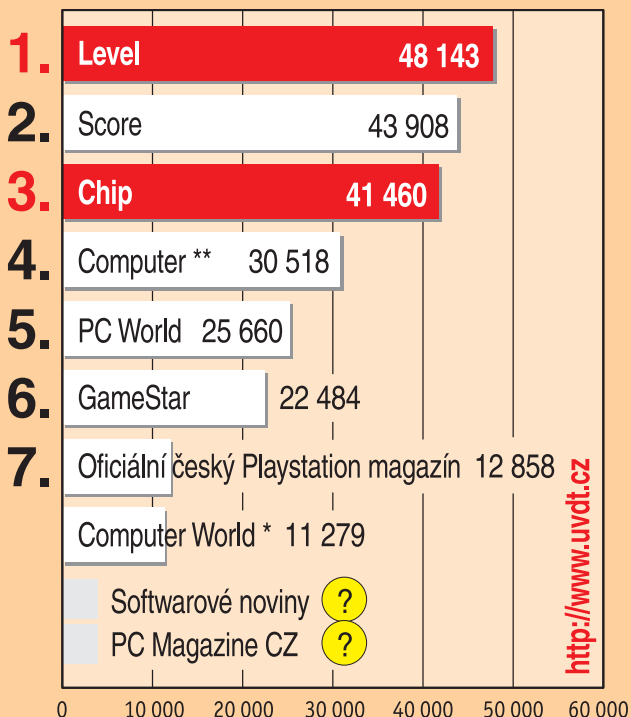
```
SELECT MAX(CIC) FROM CIN INTO
:CIX;
IF :CIX IS NULL
THEN :CIX=1;
ELSE :CIX=:CIX+1;
IF NOT EXISTS(SELECT CIC FROM CIN
WHERE NAZEV=:NX) AND :NX IS
NOT NULL
THEN INSERT INTO CIN(CIC,NA-
ZEV,CENAC) VALUES(:CIX,:NX,0.0);
END
```

Procedura ZRUSCIN ruší jen takový čin,
na který ještě nepřišlo udání:

```
CREATE PROCEDURE ZRUSCIN(CIX
INTEGER)
AS
BEGIN
IF NOT EXISTS(SELECT CICINU FROM
UDANI WHERE CICINU=:CIX )
THEN DELETE FROM CIN WHERE
CIC=:CIX;
END
```

Aktualizace názvu činu musí být domyš-
lena tak, aby nedošlo k duplicitě názvů.
Řešení tohoto problému je vidět
v proceduře NAZEVGINU:

Nejprodávanější počítačové časopisy červenec 1999



Náklady časopisů byly ověřeny agenturou ABC ČR.
? – prodaný náklad nezjistitelný, vydavatel jej nesděluje!
*týdeník, **čtrnáctideník

Národní bezpečnostní úřad

Bubenečská 20, 160 49 Praha 6

VYPISUJE NA ROK 2000 TÉMATA GRANTŮ Z ÚČELOVÝCH PROSTŘEDKŮ VÝZKUMU A VÝVOJE:

- Využití eliptických křivek v kryptografii.
- Využití neuronových sítí v kryptologii.
- Sledování nových metod ataků (např. diferenční výkonné analýzy – Differential Power Analysis) na kryptografické prostředky a nosiče informací s cílem vytvoření protiopatření.
– Přihlášky a informace na tel. čísle: **(02) 6143 1462**
- Monitorování nalezených nedostatků a rizik v operačních a komunikačních systémech, ve kterých jsou používány kryptografické prostředky, využívané státní správou.
- Návrhy metod mechanické, chemické a fyzikální ochrany nosičů informací před zneužitím po dobu jejich životnosti.
– Přihlášky a informace na tel. čísle: **(02) 6143 1371**
- Zjištění předpokládaného směru rozvoje komunikačních a informačních systémů státní správy a jejich ochrany v příštích pěti letech. Porovnání se směrem rozvoje informačních systému státní správy v členských státech NATO. Výsledky budou využity při vypracování nové koncepce ochrany utajovaných skutečností v těchto systémech.
– Přihlášky a informace na tel. čísle: **(02) 6143 1919**

Zaslání nebo vyzvednutí přihlášek je možné domluvit na uvedených telefonních číslech. Odevzdání vyplněných přihlášek, které obsahují návrhy na věcné plnění grantů včetně předpokládaných finančních nákladů je stanoven nejpozději do **17. prosince 1999.**

```
CREATE PROCEDURE NAZEV CINU
(CIX INTEGER, NX VARCHAR(30))
AS
BEGIN
IF NOT EXISTS(SELECT CIC FROM CIN
WHERE NAZEV=:NX) AND :NX IS
NOT NULL
THEN UPDATE CIN SET NAZEV=:NX
WHERE CIC=:CIX;
END
```

Procedura CENACINU umožňuje měnit sazebník poplatků udavačům na platné hodnoty:

```
CREATE PROCEDURE CENACINU(CIX
INTEGER, CX DECIMAL(10,2))
AS
BEGIN
IF :CX IS NOT NULL AND :CX >= 0.0
THEN UPDATE CIN SET CENAC=:CX
WHERE CIC=:CIX;
END
```

Poslední tři procedury se budou hodit pro aktualizaci tabulky UDANI. Tabulka má složený primární klíč a tři cizí klíče. Nechceme-li spoléhat na integritní omezení, což děláme od samého začátku, musíme vše předem otestovat:

```
CREATE PROCEDURE NOVEUDANI(DX DATETIME, UX VARCHAR(10),
OX VARCHAR(10), CX INTEGER)
AS
BEGIN
IF NOT EXISTS(SELECT DEN FROM
UDANI WHERE DEN=:DX AND
RCUD=:UX AND RCOB=:OX
AND CIGINU=:CX) AND :DX IS NOT
NULL AND :UX IS NOT NULL AND :OX
IS NOT NULL
AND :CX IS NOT NULL AND
EXISTS(SELECT RC FROM CLOVEK
WHERE RC=:UX)
AND EXISTS(SELECT RC FROM CLOVEK
WHERE RC=:OX)
AND EXISTS(SELECT CIC FROM CIN
WHERE CIC=:CX)
AND NOT(:UX=:OX)
THEN INSERT INTO (DEN,RCUD,
RCOB,CIGINU) VALUES(:DX,:UX,:OX,
:CX);
END
```

Uvedenou dřinu si můžeme ušetřit pomocí WHEN ANY, které umožní ošetřit případné chyby. Podobným trikem by bylo možné zjednodušit i některé předchozí procedury, což je necháno jako cvičení. Následuje elegantnější verze procedury NOVEUDANI:

```
CREATE PROCEDURE NOVEUDANI(DX DATETIME, UX VARCHAR(10),
OX VARCHAR(10), CX INTEGER)
AS
BEGIN
INSERT INTO (DEN,RCUD,RCOB,CIGINU) VALUES(:DX,:UX,:OX,:CX);
WHEN ANY
DO EXIT;
END
```

Někdy je třeba zahladit všechny stopy po udavači a jeho stupidní práci. Procedura ZAHLADUDAVACE nejprve zruší všechna udání, která učinil. Není-li zároveň obětí jiných udavačů, zmizí i z tabulky CLOVEK:

```
CREATE PROCEDURE ZAHLADUDAVACE(RCX VARCHAR(10))
AS
BEGIN
DELETE FROM UDANI WHERE
RCUD=:RCX;
IF NOT EXISTS(SELECT RCOB FROM
UDANI WHERE RCOB=:RCX)
THEN DELETE FROM CLOVEK
WHERE RC=:RCX;
END
```

Obdobně pomocí procedury ZAHLADOBET zmizí beze stop oběť:

```
CREATE PROCEDURE ZAHLADOBET(RCX VARCHAR(10))
AS
BEGIN
DELETE FROM UDANI WHERE
RCOB=:RCX;
IF NOT EXISTS(SELECT RCUD FROM
UDANI WHERE RCUD=:RCX)
THEN DELETE FROM CLOVEK
WHERE RC=:RCX;
END
```

Uživatelům databáze OZNAMKA stačí zveřejnit seznam názvů, významů a sloupců všech view a seznam názvů, významů a parametrů všech uložených procedur. Příslušné pohledy UDANICKO, UDAVAC, OBET, PRECIN, RCUDAV, RCOBET a MEDAILE nám umožní dokonalé výstupy informací v reálném čase a v požadovaném tvaru. Uložené procedury NOVYCLOVEK, ZRUSCLOVEK, ZMENJMENO, ZMENPRIJMENI, JAKO OBET, JAKOUDAVAC, NOVYCIN, ZRUSCIN, NAZEV CINU, CENACINU, NOVEUDANI, ZAHLADUDAVACE a ZAHLADOBET poskytnou možnost bezpečné aktualizace databáze. Jak to všechno funguje a kolik

tabulek je schováno „pod povrchem“, se nedozví ani uživatel, ani tvůrce klientské aplikace, pokud mu to někdo nevykecá. Pomocí „čínské zdi“ z procedur a view je tvrdě oddělen vnitřní systém realizující příslušné know-how analytika od vnějšího prostoru řadových realizátorů clientských aplikací a nehrozí nebezpečí „brigádnického efektu“, kdy každý inteligentní prázdninový brigádník pochopí vše potřebné ke konkurenční činnosti po prázdninách.

Slovo závěrem

Ten, kdo pozorně četl celý seriál od šedé a flekaté plastelíny v Chipu 6/98 a pochopil normalizaci tabulek, relace 1:N, integritní omezení, tabulky, domény, indexy, primární, unikátní a cizí klíče, možnosti příkazu select a výhody view a procedur, ten se jistě při samostudiu seznámí se zbylými možnostmi SQL. Rád bych v prvé řadě uvedl, proč se v seriálu nezabývám problematikami TRIGGER a GRANT, které jsou chloubou každého manuálu ke konkrétnímu SQL serveru. Ten, kdo dobře navrhl databázový systém a data aktualizuje zásadně pomocí uložených procedur, objektivně TRIGGER nepotřebuje. Ledaže by se chtěl pochlubit, že jej zná nebo se pokouší velmi moderním způsobem automaticky korigovat svůj vlastní chaos. Naproti tomu zajištění různých přístupových práv různým uživatelům pomocí GRANT je činnost pro zdárný chod aplikací na SQL serveru zcela zásadní. Na druhé straně je tak jednoduchá, že ji snadno pochopí každý, kdo dočetl až sem. Nejlépe GRANT chápou ti, kdo by nejraději všechno všem ostatním uživatelům zakázali. I zde platí všeho s mírou, a proto by zákazy neměly ochromit rutinní provoz databáze. Pokud vás právě končí seriál přivedl do rozpaků, pak splnil svůj účel. V takovém rozpoložení nejspíš něco nového dobrého vytvoříte nebo něco starého konečně předěláte tak, aby to fungovalo lépe. Až budete z klientu posílat SQL dotazy, může se stát, že neprijde kvalitní odpověď, i když je všechno dobře navrženo. Pak patrně nekladete dotaz na SQL server, ale pouze na jeho model v nějakém jiném prostředí. Za dobu svého čtyřletého SQL misionářství mi jen několik lidí vynadalo, že jsem je přivedl ke špatné víře. Ve skutečnosti nepracovali s daty na SQL serveru, ale se soubory ve formátu DBF či MDB. Jejich potíže se týkaly sdílení dat nebo práce s velkým množstvím položek v jedné tabulce.

JAROMÍR KUKAL