

Triggery

V našem databázovém systému potřebujeme kontrolovat, jestli je v každém registrovaném týmu právě tolik koní a tolik lidí, kolik je v dané soutěžní kategorii povoleno. To se ukázalo jako netriviální omezení, které je potřeba kontrolovat triggerem. Rozhodli jsme se, že se počty zkontrolují ve chvíli, kdy se má přiřadit tým k závodu. Tím se uzavře registrace týmu a již dále nelze měnit jeho obsazení. Samotný trigger je ve výpisu 1...

Kód 1: trigger check pocty.

```
1 CREATE OR REPLACE FUNCTION check_pocty() RETURNS TRIGGER AS '  
2     DECLARE  
3         pocet_koni_povoleny INTEGER;  
4         pocet_lidi_povoleny INTEGER;  
5         pocet_koni_realny INTEGER;  
6         pocet_jezdcu INTEGER;  
7         pocet_lidi_realny INTEGER;  
8         check_tym_id INTEGER;  
9     BEGIN  
10  
11     IF (TG_OP = 'INSERT') AND NEW.zavod_id IS NOT NULL THEN  
12         RAISE EXCEPTION  
13             'Tým lze přiřadit k závodu až po jeho naplnění!';  
14     END IF;  
15     IF NEW.zavod_id IS NULL THEN  
16         RETURN NEW;  
17     END IF;  
18     check_tym_id := NEW.tym_id;  
19     SELECT pocet_koni, pocet_přisedicich  
20         INTO pocet_koni_povoleny, pocet_lidi_povoleny  
21         FROM kategorie  
22         INNER JOIN tymy ON kategorie.kategorie_id = tymy.kategorie_id  
23         WHERE tymy.tym_id = check_tym_id;  
24     SELECT COUNT(kone_kun_id) INTO pocet_koni_realny  
25         FROM tymy_has_kone GROUP BY tymy_tym_id  
26         HAVING tymy_tym_id = check_tym_id;  
27  
28     SELECT COUNT(osoby_osoba_id) INTO pocet_lidi_realny  
29         FROM tymy_has_osoby WHERE NOT je_jezdec GROUP BY tymy_tym_id  
30         HAVING tymy_tym_id = check_tym_id;
```

```

31     SELECT COUNT(osoby_osoba_id) INTO pocet_jezdcu
32         FROM tymy_has_osoby WHERE je_jezdec GROUP BY tymy_tym_id
33         HAVING tymy_tym_id = check_tym_id;
34
35     IF NOT pocet_koni_povoleny=pocet_koni_realny
36         OR pocet_koni_realny IS NULL THEN
37         RAISE EXCEPTION 'Spatny pocet koni!';
38     ELSEIF NOT pocet_lidi_povoleny=pocet_lidi_realny
39         OR pocet_lidi_realny IS NULL THEN
40         RAISE EXCEPTION 'Spatny pocet prisedicich!';
41     ELSEIF NOT pocet_jezdcu = 1 OR pocet_jezdcu IS NULL THEN
42         RAISE EXCEPTION 'Spatny pocet jezdcu!';
43     END IF;
44
45     RETURN NEW;
46 END
47 ,
48 LANGUAGE plpgsql;
49
50 CREATE TRIGGER trig_pocty
51 BEFORE UPDATE OR INSERT ON tymy
52 FOR EACH ROW
53 EXECUTE PROCEDURE check_pocty();

```

Uložené procedury

V naší databázi používáme uloženou proceduru `prirad_startovni_cisla(zavod_id)` (ve výpisu 2) která přiřadí všem týmům zaregistrovaným v daném závodě náhodná startovní čísla od jedné do počtu týmů. Nejprve jsme k tomuto účelu chtěli použít SEQUENCE, ale ukázalo se, že to není praktické řešení. Naše procedura tedy nejdříve SELECTuje všechny týmy a náhodně je seřadí a tyto výsledky pak prochází po řádkách a do startovního čísla ukládá postupně inkrementovanou lokální proměnnou.

Kód 2: Uložená procedura `prirad_startovni_cisla`

```

1 CREATE OR REPLACE FUNCTION prirad_startovni_cisla(zavod INTEGER) RETURNS ↵
   boolean AS '
2     DECLARE
3         i RECORD;
4         update_count INTEGER;
5         cislo INTEGER;
6     BEGIN
7         cislo := 1;
8         FOR i IN SELECT tym_id FROM tymy WHERE zavod_id = zavod ORDER BY ↵
           random() LOOP

```

```

9          RAISE DEBUG ''updating id %'', i.tym_id;
10         UPDATE tymy SET startovni_cislo = cislo WHERE tym_id = i.↵
           tym_id;
11         GET DIAGNOSTICS update_count = ROW_COUNT;
12         cislo := cislo+1;
13         IF update_count < 1 THEN
14             RETURN FALSE;
15         END IF;
16     END LOOP;
17     RETURN TRUE;
18 END
19 ,
20 LANGUAGE plpgsql;

```

Statistiky

Česká Jezdecká Federace každoročně vyhlašuje jezdce, který strávil na kolbišti nejvíce času. Pro tuto statistiku jsme vytvořili následující dotaz (ve výpisu 3)

Kód 3: Statistika času na kolbišti.

```

1 SELECT osoby.jmeno, osoby.prijmeni, vnitrek.cas FROM osoby
2 INNER JOIN (
3     SELECT osoby.osoba_id,
4     SUM(cas_prvniho_kola+cas_druheho_kola) AS cas FROM osoby
5     INNER JOIN tymy_has_osoby ON
6         osoby.osoba_id=tymy_has_osoby.osoby_osoba_id
7     INNER JOIN tymy ON tymy_has_osoby.tymy_tym_id=tymy.tym_id
8     INNER JOIN zavod ON zavod.zavod_id=tymy.zavod_id
9     WHERE tymy_has_osoby.je_jezdec
10    AND EXTRACT(year from zavod.datum)=2014
11    GROUP BY osoby.osoba_id
12    ORDER BY cas DESC) AS vnitrek ON osoby.osoba_id=vnitrek.osoba_id;

```
