Databázové systémy

Dokumentace projektu

Autoři:

Zdeněk Brhel - xbrhel04
Petr Kašpar - xkaspa40

Datum:
28. 4. 2018

1. Část projektu

Zadání

Navrhněte informační systém pro organizaci kriminální činnosti a setkání předních brněnských rodin. V čele každé rodiny je Don (např. Don Calzone, Don Quatro Formagi, Don Funghi, apod.), respektovaná osoba, která organizuje dění rodiny, určena běžnými atributy (jméno, věk, velikost bot, ???). Samotná famílie pak sestává z řadových členů, kteří nemusí být pokrevně spřízněni a kteří zprostředkovávají veškerou kriminální činnost, jako je odstraňování nevhodných osob, dealování drog, tisknutí falešných peněz, atd., Don si však nikdy nešpiní ruce. Kriminální činnost probíhá v rámci území, určených gps souřadnicemi, adresou, rozlohou, a mohou být buď řádným rajónem jedné z familií, nebo neutrálním územím, kde může operovat větší počet familií. Do samotné kriminální činnosti jsou pak zapojeni konkrétní řadoví členové v různých obdobích (tzn. řadový člen může nejprve dealovat drogy a poté přejít na odstraňování osob), přičemž každá činnost je přímo vedena jednou z familií a má různou dobu trvání a tato operace má vždy unikátní jméno. Do těchto činností může být zapojeno více rodin, to v případě, že dvě rodiny uzavřou alianci, které však nemusí mít dlouhé trvání. Jednou z hlavních činností jsou pak vraždy, u kterých je vedeno nejen, kde byly provedeny, ale i kdo je provedl a na kom je provedl. Některé z těchto vražd jsou pak objednány samotnými Dony, jiné jsou spontánní, bez bližší objednávky. Cílem vraždy však nikdy není Don. Systém pravidelně organizuje setkání Donů, kde se probírají výsledky jednotlivých rodin a probíhají buď v rámci neutrálního území, nebo jsou hostěny některou z předních rodin. Systém pak zasílá Donům pozvánky a očekává od nich potvrzení účasti.

2. - 5. Část projektu

Implementace databáze

Skript vytvoří základní objekty databáze, přičemž nastaví omezení (klíče..). Následně pak nahraje do databáze hodnoty; vykonává různou manipulaci s daty a demonstruje funkcionalitu výše zmíněných omezení. Diagramy a schéma databáze se nachází na konci dokumentu.

Triggery

V souladu se zadáním jsme vytvořili dva databázové triggery, přičemž první z nich byl implementován již ve druhé fázi projektu. První trigger kontroluje formát rodného čísla v tabulce Clen (RC_Clena). Rodné číslo je testováno na následující: musí být posloupností šesti číslic s lomítkem následováno třemi (RC do roku 1953) nebo čtyřmi číslicemi. Dále musí být splněna podmínka, že dvojčíslí značící den narození musí být v rozsahu 1 až 31. Dvojčíslí značící měsíc musí být v rozsahu 1 až 70. Pokud je délka RC rovna 10, nesmí být rok větší než 53. Pokud je délka rovna 11, kontroluje se dělitelnost jedenácti. V triggeru je použita vyjímka (EXCEPTION) na VALUE_ERROR kvůli konverzi rodného čísla z VARCHAR na NUMBER. Druhý trigger obsluhuje vkládání do tabulky Familie. Pokud je ID_Familie rovno NULL, bude ze sekvence zvolena číslice a použita jako ID_Familie. Hodnota inkrementace je nastavena na 1. Protože sekvence na serverech Oracle nejsou gap – free, nelze zaručit, že čísla generována sekvencí budou

nutně inkrementována o nastavenou hodnotu, tedy mohou se vyskytnout tzv. gaps, tedy čísla generována po sobě nemusí tvořit posloupnost. Oba triggery se spouští před vložením do příslušné tabulky – BEFORE INSERT.

Procedury

Dle zadání jsme měli vytvořit dvě netriviální procedury obsahující použití kurzoru, ošetření vyjímek a proměnnou s odkazem na řádek nebo sloupec tabulky. Procedura *percentMurder(id_Uzemi)* spočítá počet spáchaných kriminálních činností na území zadaném v parametru procedury a procentuální zastoupení vražd. Procedura obsahuje kurzor, vyjímku na dělení nulou a použití proměnné odkazující se na řádek tabulky. V případě dělení nulou je vyvolána chyba a vypsáno hlášení o chybě. Druhá procedure *printMurderers()* obsahuje dva kurzory – jeden pro kriminální činnosti a druhý pro vraždy. Ve dvourozměrném cyklu jsou porovnávány hodnoty ID_Cinnosti z obou kurzorů a pokud se rovnají a zároveň je typ činnosti “Vražda”, tak je vypsáno jméno a rodné číslo vraha.

Explain plan

Demonstrovaný na jednoduchém dotazu:

```
SELECT JmenoFamilie,
to_date(round(avg(to_number(to_char(DatumNarozeni, 'J')))), 'J')
AS "Prumerne datum narozeni"
  FROM Familie NATURAL JOIN Clen
  WHERE JmenoFamilie = 'Corleone'
GROUP BY JmenoFamilie;
```

Bez indexu

Nejprve jsme EXPLAIN PLAN (plán, jak databáze zpracovává daný dotaz) spustili bez použití indexu. Výstupem bylo:

ID	Operaton	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		2	104	6 (0)	00:00:01
1	SORT GROUP BY NOSORT		2	104	6 (0)	00:00:01
2	HASH JOIN		2	104	6 (0)	00:00:01
3	TABLE ACCESS FULL	FAMILIE	1	30	3 (0)	00:00:01
4	TABLE ACCESS FULL	CLEN	6	132	3 (0)	00:00:01

S indexem

Zvolený index byl na tabulce FAMILIE na sloupci JmenoFamilie, protože se dá očekávat, že se k těmto položkám bude přistupovat velmi často. S indexem následný EXPLAIN PLAN vypadal takto:

ID	Operaton	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		2	104	5 (0)	00:00:01
1	SORT GROUP BY NOSORT		2	104	5 (0)	00:00:01
2	HASH JOIN		2	104	5 (0)	00:00:01
3	TABLE ACCESS BY INDEX ROWID	FAMILIE	1	30	2 (0)	00:00:01
4	INDEX RANGE SCAN	MYINDEX	1		1 (0)	00:00:01
5	TABLE ACCESS FULL	CLEN	6	132	3 (0)	00:00:01

Rozebrání

SELECT STATEMENT – Uskutečnění dotazu SELECT
SORT GROUP BY NOSORT – Není požadován žádný SORT, GROUPne řádky podle hodnot
HASH JOIN – Provede se příkaz JOIN podle hashovacího klíče
TABLE ACCESS FULL – Prochází se celá tabulka (bez indexu)
TABLE ACCESS BY INDEX ROWID – Přistupuje do tabulky přes konkrétní řádek (s indexem)
INDEX RANGE SCAN – Přistoupí k vedlejším položkám indexu a použije hodnotu v indexu pro získání odpovídajících řádků

Přidělení práv

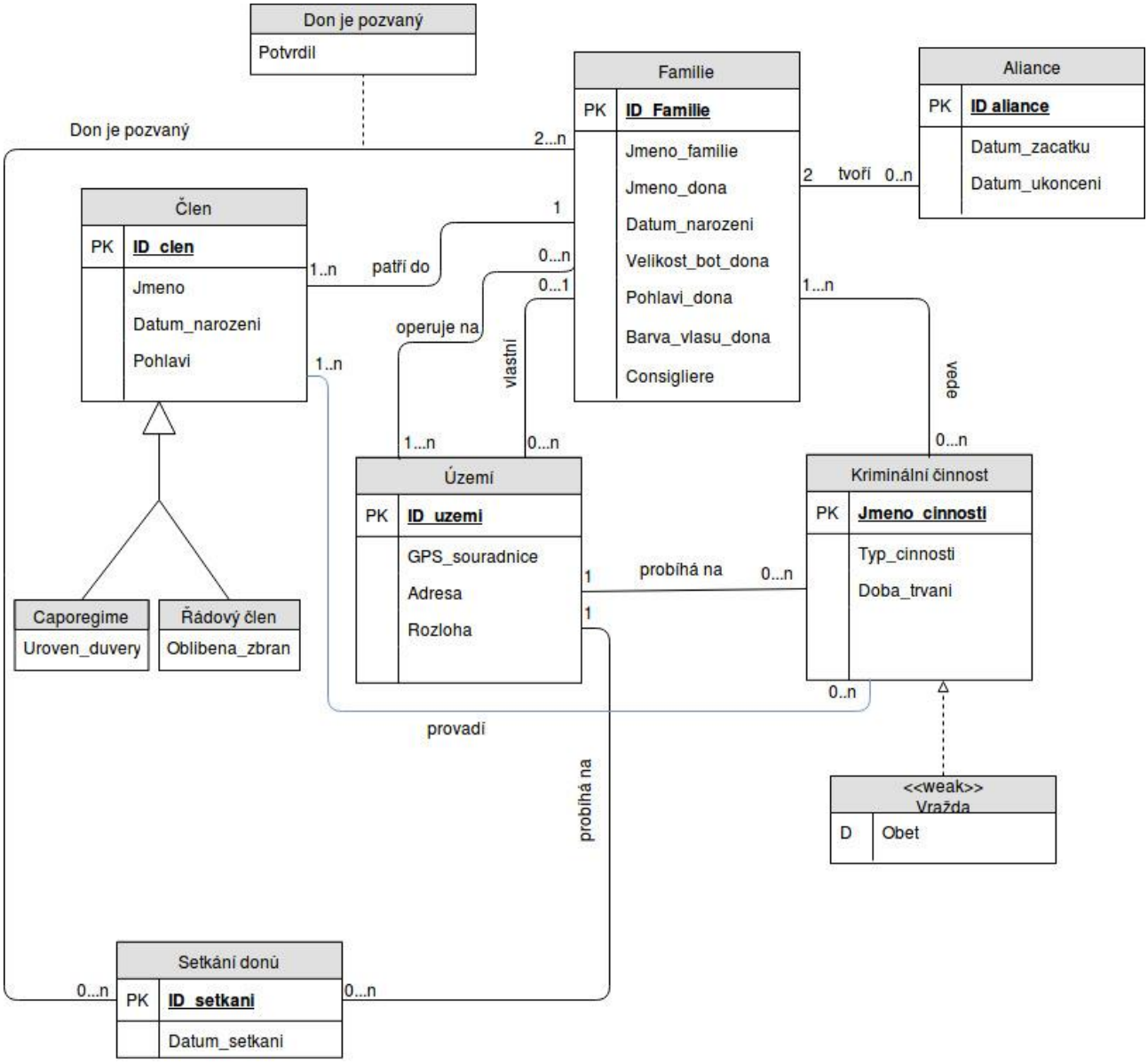
Přidělení práv se řeší příkazem `GRANT {PRÁVO} ON {CO} TO {KOMU}`, kde {PRÁVO} je jaká přístupová práva se uživateli dávají (ALL pro všechny práva, EXECUTE na spouštění, ..), {CO} je název objektu/databáze/… a {KOMU} je název uživatele.

Materialized view

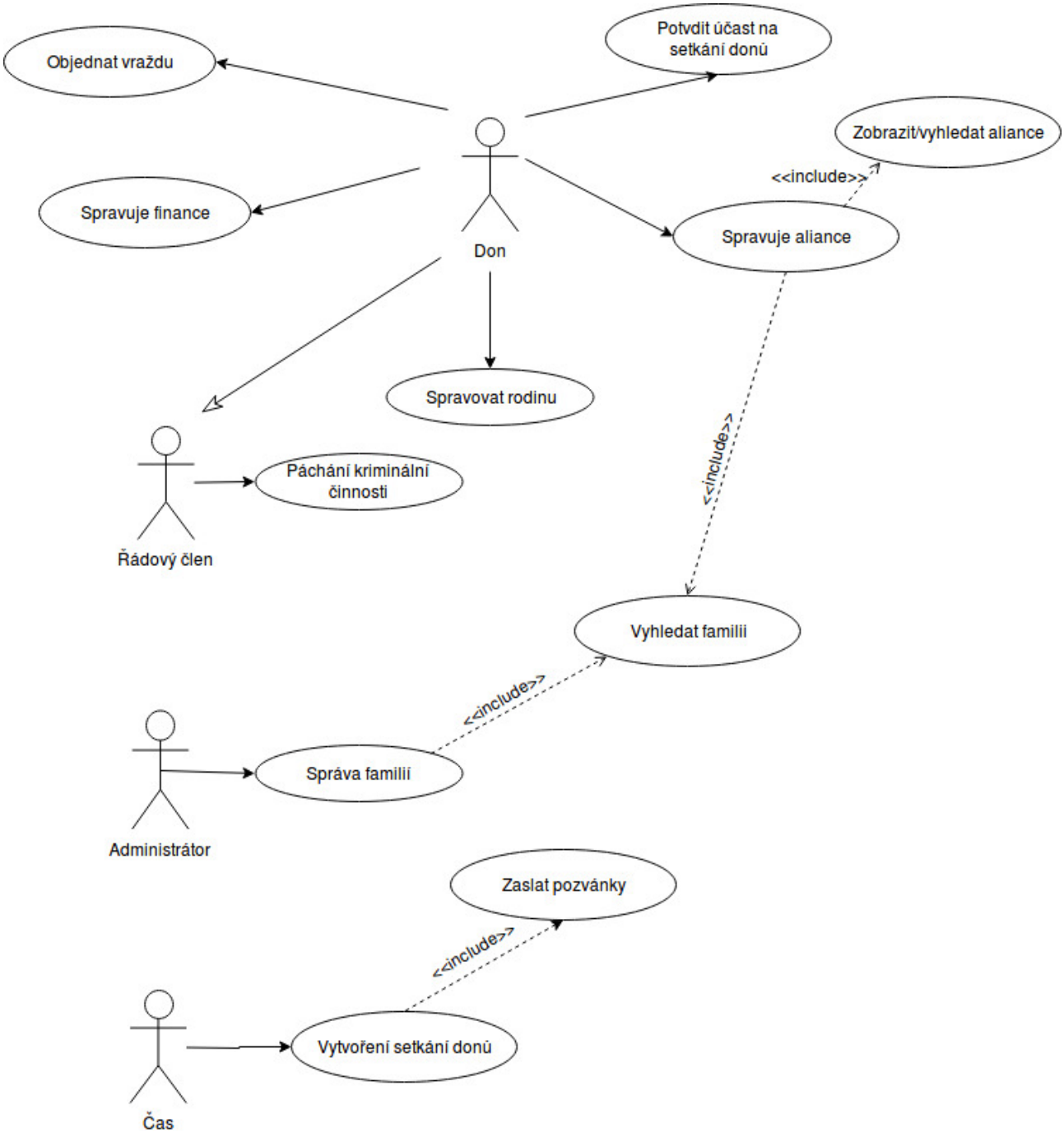
Materializovaný pohled je nastavený, aby se znovu načetl při COMMITu. BUILD IMMEDIATE zde není napsaný, avšak je to výchozí nastavení, proto není nutné ho psát. Způsobuje to, že hned jak se materializovaný pohled vytvoří, tak se i naplní.

Aktualizaci materializovaného pohledu jsme si hned vyzkoušeli tím, že jsme přidali do tabulky další záznam, použili příkaz COMMIT a znovu zobrazili tento pohled.

Příloha 1 – ER diagram



Příloha 2 – Use-Case diagram



Příloha 3 – schéma databáze