dbMan – modulární SQL konzole

Milan Šorm

Ústav informatiky
Provozně ekonomická fakulta
Mendelova zemědělská a lesnická univerzita v Brně
Email: sorm@pef.mendelu.cz

Abstrakt: Snahou všech datamanažerů (ať již DBA, vývojářů či datamanažerských „lopat“) je najít nástroj, který bude minimálně překážet při výkonu každodenních starostí a přitom poskytne maximální komfort v možnostech a případném dalším individuálním rozšiřování. Tyto tendence mne vedly k návrhu a vytvoření open source SQL konzole, která je striktně modulární a snaží se splnit výše uvedené podmínky. Tento příspěvek je o návrhu tohoto nástroje, o úskalích, které Vás mohou potkat a o metodách, které Vám mohou pomoci při vývoji podobných modulárních nástrojů pro jiné oblasti. Současně příspěvek představí možnosti této (zatím jen) textové konzole a ukáže, kudy se ubírá její další vývoj.

Klíčová slova: dbMan, SQL konzole, modularita, Perl.

1 Úvod

Pracuji s databázovými systémy už řadu let, ale teprve před čtyřmi roky jsem je začal využívat jako centrální úložiště všech dat. Od roku 1998 vyvíjíme na naší univerzitě informační systém, který si klade velmi ambiciózní cíle – a jedním z nich je být centrálním datovým skladem univerzity. To ovšem vyžaduje vznik pro univerzitu zcela nové profese – datamanažerů. Jako první datamanažer nového informačního systému naší univerzity jsem na jaře roku 1998 hledal nástroj, který by splňoval všechny mé požadavky – především jednoduchou obsluhu, mnoho nástrojů pro odstranění každodenní dřiny, rozsáhlou modularitu pro možnost doplňovat si nové funkce a jednoduché (případně modulární) rozhraní, aby bylo možné s nástrojem pracovat i na pomalých českých telefonních linkách.

Bohužel většina open source i komerčních SQL konzolí v té době byla určena vždy pro jeden konkrétní databázový systém a neposkytovala komfort, který jsem od nástroje očekával. Zvolil jsem proto vlastní cestu (a od té doby velmi často volím vlastní cestu) a vyvinul si jednoduchou SQL konzoli v Perlu využívající DBI/DBD rozhraní, které umožňuje práci s celou řadou databázových systémů.

Tento nástroj jsem vyvíjel zhruba jeden rok, než se z něj stal použitelný nástroj splňující některé z uvedených požadavků – především obsahoval řadu nadstandardních funkcí a zárodek modularity. Až do roku 2002 zůstal nástroj

Jan Kasprzak, Petr Sojka (editoři): SLT 2002 – sborník semináře o Linuxu a TEXu, str. 1–8, 2002.

stabilní (s minimální opravou chyb) a byl dostupný v podobě samostatného balíčku i přes vyhledávač `freshmeat.net`. Na jaře roku 2002 jsem se rozhodl pro kompletní přepis původního částečně monolitického balíku do plně modulárního nástroje, který bude mít jedinou filozofii „vše bude nahraditelné za pomoci modularity“. Tak vznikl dnešní dbMan, který jako rozšiřovací modul implementuje i příkaz `QUIT`.

2 Instalace nástroje

Současný dbMan je nalezitelný na síti CPAN (Comprehensive Perl Authors Network) a lze jej jednoduše instalovat pomocí cpan shellu. Pro instalaci použijte následující postup:

```
perl -MCPAN -e shell
cpan> install DBIx::dbMan
```

Pro správnou funkčnost vyžaduje dbMan přítomnost interpretu jazyka Perl (nezáleží na operačním systému, já provozuji dbMan pod Linuxem, Solarisem i MS Windows) a moduly `DBI`, `Data::ShowTable`, `Text::FormatTable`, `Text::CSV`, `Term::Size`, `Term::ReadLine` a samozřejmě příslušné ovladače `DBD` k Vaší databázi. Pro provoz alternativního grafického rozhraní (které je silně ve vývoji) je nutná distribuce `Perl/Tk`. Přímé závislosti nainstaluje cpan shell za Vás při instalaci dbManu, na instalaci ostatních modulů použijete s výhodou opět cpan shell.

Součástí instalované distribuce je nejen vlastní program (ve jmenném prostoru `DBIx::dbMan::`), ale také jednoduchá dokumentace popisující jak začít dbMan rozšiřovat a asi 70 základních rozšíření poskytující standardní funkčnost dbMana (z těchto jednoduchých kódů je možné vyjít při pokusu o rozšíření). Rovněž je přítomen patch pro `Term::ReadLine::Gnu`, který opravuje chybu znemožňující používat TAB kompletaci bez rozlišení velikosti písmen.

3 Možnosti SQL konzole

SQL konzole je nástroj pro přímou komunikaci s databázovým systémem. Umožňuje zasílat do DBS příkazy a rozumně zobrazovat odpovědi serveru. Tyto zasílané příkazy lze rozlišit na dvě velké skupiny – dotazy (vrací tabulku u relačních databázových systémů) a úkoly (vrací informaci o splnění úkolu, příp. počet záznamů u relačních databázových systémů, které úkol ovlivnil).

I při takto jednoduchém úkolu lze však nalézt řadu oblastí, kde nám může SQL konzole usnadnit život – ať již se jedná o historii odesílaných příkazů nebo použití tabelátoru pro dohledávání názvů tabulek, sloupců a dokončování příkazů.

dbMan rozděluje možnosti do dvou kategorií – možnosti rozhraní SQL konzole a možnosti vnitřního interpretu příkazů (vykonávání akcí uživatele). To umožňuje oddělit formu vzniku akce (zadání příkazů v příkazovém prostředí, výběr z menu, kliknutí myši na ikoně) od její interpretace (provedení příkazu v dbManu, databázi, transformace akce na jinou akci apod.).

3.1 Rozhraní konzole

V současně době existují dvě rozhraní pro práci s dbManem. Rozhraní `cmdline` je standardním, plně funkčním, denně používaným rozhraním, jehož filozofií je minimalizace přenášených dat mezi serverem, kde běží dbMan a terminálem, kde pracuje uživatel (ideální pro pomalé telefonní linky, u nás však díky zvyku používané i na Gigabitové páteřní síti univerzity). Toto rozhraní nabízí práci v terminálovém režimu, kdy na výzvu (prompt) načte uživatelův příkaz a odešle jej do jádra dbManu na zpracování jako akci `COMMAND`. Na druhou stranu nabízí jednoduchou metodu pro výstup informací na terminál (`print()`), jenž může být později použita rozšířením realizující výstup. Pokud rozhraní nalezne v systému knihovnu `Term::ReadLine`, použije její vlastnosti pro načítání a úpravy příkazového řádku, její vlastnosti pro práci s historií příkazů (které doplní o možnost transparentní historie ukládáním historie i do souboru) a její vlastnosti pro tabelátorovou kompletaci (doplňování slov při stisku klávesy TAB). Vlastní implementace tabelátorové kompletace je však otázkou rozšiřujících modulů. Příkaz `dbman` nastartuje dbMan v tomto rozhraní.

Druhým rozhraním je experimentální rozhraní `tkgui`, které ukazuje sílu rozšiřujících možností dbManu zavedením grafické konzole v podobě okna s příkazovým řádkem a výstupním prostorem. Toto rozhraní (stejně jako předchozí) implementuje načítání příkazů a výstup výsledku, ovšem tentokrát je výstup směřován do okna aplikace užívající `Perl/Tk`. Rozhraní implementuje vlastní ovládání historie příkazů kompatibilní s předchozím rozhraním, nepodporuje však zatím TAB kompletaci. Všechny možnosti dbManu pracují v obou rozhraních, protože se liší pouze forma vzniku akcí a zobrazování výstupů. Toto rozhraní získáme příkazem `xdbman`.

3.2 Interpret akcí

Interpret akcí zajišťuje předávání vzniklých akcí do jednotlivých rozšiřujících modulů a tím zajišťuje vlastní funkčnost dbMana. Pomocí těchto modulů realizuje dbMan veškerou funkčnost – počínaje reakcí na příkaz `QUIT`, předávání SQL příkazů do databáze, implementaci TAB kompletace, práci se schránkou, práci s historií příkazů apod.

Z funkcí dbManu, které datamanažeři pracující na naší univerzitě nejvíce chválí, bych vybral především možnost paralelního spojení s více databázovými systémy (pomocí příkazu `CREATE CONNECTION` můžeme zakládat další spojení na různé databázové systémy, pomocí příkazu `USE` pak mezi těmito spojeními přepínáme). Pokud spojíme další pěknou vlastnost – práci se schránkou SQL výstupů (clipboardem) – s vícenásobnými spojeními, dostaneme jednoduchý nástroj pro datamanažerské pumpování dat mezi různými systémy (v jedné databázi – např. v DBF souboru – provedeme pomocí `\copy` vybrání nějakého výstupu SQL dotazu, v druhé databázi – např. v Oracle – pomocí `\paste` výstup jednoduše zapíšeme do požadované tabulky). Při použití `\unioncopy` pak můžeme nasbírat i více výstupů z různých zdrojů a výsledek poté zapsat do jiné tabulky (např. CSV souboru nebo do tabulky databáze na MS SQL serveru).

Samozřejmá je podpora transakcí (transakce v jednom spojení neovlivňují ostatní spojení), výpočtu doby pro zpracování příkazu (jednoduchý benchmark), na databázi Oracle zobrazení stromu `EXPLAIN PLAN`, zobrazení seznamu objektů v databázi nebo popisu (`DESCRIBE`) tabulek. Víceřádkové SQL dotazy, možnost stránkování výstupu programem `less`, ukládání výstupu do souboru či interpretace `.sql` souborů je už dnes považována za samozřejmost. Datamanažeři oceňují i jednoduchý vstupní CSV filtr, který na základě popisu oddělovačů v textovém souboru provede přenos tohoto souboru do databáze (podobnou věc lze realizovat pomocí schránky, ale potřebujeme ovladač `DBD::CSV` pro práci s takovými soubory, což jednoduchý filtr `\csvin` nepotřebuje).

Naši datamanažeři často editují v databázi Oracle objekty PL/SQL kódu (procedury, funkce, triggery, package), k čemuž dosud využívali především DBA studio (standardní Oracle GUI nástroj napsaný v Javě). Tento nástroj je nepoužitelný nejenom na telefonní lince, ale často i na naší rychlé univerzitní síti. Proto jsem dbMan rozšířil o možnost (zatím jen pro Oracle) přímé editace takovýchto objektů – ve variantním editoru pak můžete dosáhnout dalších schopností, např. při použití editoru `vim` získáte zdarma barevnou syntaxi PL/SQL, což neumí ani DBA studio).

Poslední vítané vlastnosti dbManu, kterou si zde představíme, je podpora různých výstupních formátů a podpora TAB kompletace. První vlastnost umožňuje pomocí rozšíření definovat výstupní formát SQL dotazů (tabulka, CSV, HTML, speciální `records` formát pro malé dotazy apod.), kdy lze definovat odlišný formát pro jednořádkové výstupní tabulky a ostatní dotazy. TAB kompletace je plně řízená pomocí rozšíření, takže přidání rozšíření nejen doplní on-line nápovědu programu, ale také doplňování příkazů o nové možnosti. TAB kompletace také implementuje doplňování názvů tabulek, pohledů, sekvencí a sloupců v tabulkách. Pro Oracle navíc obsahuje řadu podpůrných funkcí pro urychlení TAB kompletace užitím systémového katalogu Oracle.

4 Architektura dbManu

Vnitřní architektura dbManu se snaží dodržet striktně modulární koncepci. Základem je tzv. dbMan core (tedy jádro programu), které obsahuje tzv. candidate select algoritmus pro výběr, která rozšíření jsou v dosahu (a vybere nejvhodnější a nejnovější verze) a algoritmus pro zpracování akcí (základní smyčku získej akci – zpracuj akci obohacenou o provádění akcí v rozšířeních, jak bude popsáno níže). Toto jádro pracuje s celou řadou dalších modulů, především s dbMan interfaces moduly (rozhraní), které mohou implementovat odlišné vizuální rozhraní dbManu (jako již zmiňované `cmdline` či `tkgui`) a s dbMan DBI modulem, který zapouzdřuje práci s DBI/DBD tak, aby fungovalo paralelní spojení s více databázovými systémy. Pro celý dbMan i všechna rozšíření se dbMan DBI chová jako standardní DBI obohacené o funkce pro definice, zavádění, rušení a přepínání jednotlivých spojení.

Nejzajímavější částí, se kterou dbMan core spolupracuje, jsou dbMan extensions moduly – což jsou jednotlivé definice rozšíření. Tyto moduly samozřejmě

využívají dbMan interfaces, dbMan DBI i samotné jádro. Aby mohla jednotlivá rozšíření spolupracovat, využívají dbMan MemPool modul, který obsahuje společný datový sklad pro všechna rozšíření. Dva pomocné moduly – dbMan history a dbMan config jsou určeny pro práci s historií příkazů v souboru (transparentnost historie při různých spuštěních dbMana) a pro práci s konfiguračním souborem (čímž se pro rozšíření nabízí možnost získávat informace o svém provozu ze společné konfigurace programu dbMan).

Nejzajímavější částí architektury je mechanismus zpracování událostí a předávání akcí. Tento mechanismus (implementovaný v dbMan core) pracuje na principu následujícího algoritmu:

1. Zjištění události pomocí rozšíření.
2. Vytvoření akce (speciální asociativní pole pro akce).
3. Výběr rozšíření s maximální prioritou.
4. Předání vytvořené akce do vybraného rozšíření.
5. Rozšíření se rozhodne, zda jde o akci určenou pro něj.
6. Pokud není akce určená pro něj, vybere se další rozšíření v pořadí a přejde se k bodu 4.
7. Pokud je akce určená pro něj, vykoná se příslušná akce.
8. Dále se transformuje asociativní pole akce, typ akce a dojde k nastavení speciálního příznaku ovlivňujícího další chod zpracování akce.
9. Pokud je příznak nastaven, přejde se s upravenou akcí do bodu 3.
10. Pokud není příznak nastaven, pokračuje se výběrem další akce podle priorit a přejde se k bodu 4.
11. Celý mechanismus končí v okamžiku, kdy žádné rozšíření již nechce na akci reagovat.
12. Pokud zbylá akce je akcí `QUIT`, ukončí se celá hlavní smyčka, jinak se přejde k bodu 1.

Tento algoritmus umožňuje zpracování akcí v rozšířeních a předávání si informací mezi jednotlivými rozšířeními. Každé rozšíření také může ovlivnit, zda se má transformovaná akce znovu spustit přes všechna rozšíření, nebo se bude pokračovat dál podle pořadí priorit. Každé rozšíření také samo definuje, s jakou prioritou mu má být akce předávána (a tím ovlivňuje okamžik, kdy dostane akci ke zpracování – nová implementace dané funkce pak může mít vyšší prioritu než původní modul a tak mít možnost akci ovlivnit o něco dříve).

Ukázku toho, jak komplexní může být zpracování jednoho příkazu `SELECT` dává následující posloupnost zpracování akcí a rozšíření, které na ní reagují (přepis není úplný, vypisuji zde pouze podstatné akce a jejich úkol).

Nejprve vzniká akce `COMMAND`, která obsahuje informaci o tom, jaký příkaz byl uživatelem zadán. Pokud vynecháme preprocesor příkazů doplňující či odstraňující koncový středník podle vlastností databázového systému a ovlivňující formát výstupu, dospěje akce do rozšíření `CmdStandardSQL`, které rozpozná SQL dotaz a převede akci na akci `SQL`, která navíc obsahuje informaci o tom, zda se jedná o dotaz či úkol. Tato akce dále postupuje posloupností priorit, až dospěje do rozšíření `StandardSQL`, které provede příslušnou operaci pomocí dbMan DBI a neformátovaný a neupravovaný výstup přepíše do akce `SQL_RESULT`. Tato akce dospěje do

`SQLShowResult`, kde je podle typu SQL proveden buď výstup informace o počtu ovlivněných řádků, nebo transformace na akci `SQL_OUTPUT`, která postupuje přes řadu modulů provádějících transformace výsledku (`NULL` hodnoty, nezobrazitelné znaky, rozhodování o formátování) až do rozšíření `SQLOutputTable`, což je jedno z rozšíření realizujících konkrétní výstupní formát (takových rozšíření je celá řada). Toto rozšíření přeformátuje tabulku pomocí `Text::FormatTable` a produkuje akci `OUTPUT`, která obsahuje informaci o tom, co má být součástí výstupu. Tato akce může být provedena řadou rozšíření realizujících stránkování, uložení výstupu do souboru apod., ale pravděpodobně dospěje do rozšíření `Output`, které využije dbMan interfaces pro zobrazení výsledku a transformuje akci na `NONE`. O tuto akci už nikdo nemá zájem (ani rozšíření `Fallback`, které zobrazuje nápisy jako `Unknown command.` či `INTERNAL ERROR: Action not handled.`) a tak je cyklus zpracování události ukončen.

5 Jak dbMana rozšiřovat

Rozšíření se píší v jazyce Perl jako standardní moduly ve jmenném prostoru `DBIx::dbMan::Extension::`, přičemž tyto moduly se umísťují buď do aktuální adresáře, nebo tam, kam míří `extensions_dir` direktiva v konfiguračním souboru. Tento modul realizujeme jako potomka modulu `DBIx::dbMan::Extension` a musíme přetížit minimálně funkci `IDENTIFICATION()`, pomocí které se rozlišuje, která verze rozšíření je nejnovější, a které moduly implementují stejné funkce. Jedná se o trojici šesticiferných údajů, přičemž první část udává autora, druhá část modul v rámci autora a poslední verzi (např. `999999-000001-000005` představuje pátou verzi prvního experimentálního modulu – protože tento autor je definován jako experimentální).

Aby rozšíření bylo k něčemu dobré, je vhodné přetížit také funkci `handle_action()` realizující vlastní výkonný kód a funkci `preference()`, která definuje prioritu rozšíření pro seznam rozšíření (jak brzo dostane modul akci k provádění). Preference rozdělujeme do několika kategorií – standardně je preference 0, ale fallback a output rozšíření mají preferenci zápornou. Výkonné moduly (provádění SQL apod.) mají preferenci pod 1000, parsery jednotlivých příkazů pak pod 2000. Preprocesory (přepínače apod.) preferenci pod 3000. Makroprocesory (druhá abstrakce nad rozšířeními) pak ještě vyšší priority – až po 4000. Nad 4000 jsou preference tzv. URGENT rozšíření, např. odstranění počátečních a koncových mezer celého příkazu v akci `COMMAND` apod.

Základní kostra rozšíření je následující:

```
package DBIx::dbMan::Extension::Název;
use strict;
use vars qw/$VERSION @ISA/;
use DBIx::dbMan::Extension;

$VERSION = '0.01';  @ISA = qw/DBIx::dbMan::Extension/;
1;
```

```
sub IDENTIFICATION { return "999999-000001-000001"; }

sub preference { return 50; }

sub handle_action {
        my ($obj,%action) = @_;
        # %action modification or something making
        $action{processed} = 1;  return %action;
}
```

Úkolem `handle_action()` je zpracovat `%action` (předeším typ akce je v `$action{action}`) a toto asociativní pole modifikovat a nastavit (nebo nenastavit) příznak `$action{processed}`. Vše odpovídá výše popsanému algoritmu. Je velmi vhodné směřovat všechny výstupy (i chybové) do akce `OUTPUT`, protože jedině tímto způsobem dosáhneme nezávislosti na rozhraní. Další konvencí je akce `NONE`, na kterou by nikdo neměl reagovat. Pro sledování průchodu rozšířeními (ladění předávání akcí) lze využít trasovacího režimu (`SET TRACING ON`).

Uvnitř rozšíření lze použít pět základních objektů pomocí `$obj->{-název}`, kde objekt `-interface` zpřístupňuje aktuální rozšíření, objekt `-dbi` rozhrání dbMan DBI, objekt `-mempool` sdílený datový sklad všech rozšíření (např. předávání informace o tom, jaké výstupní formáty existují apod.), objekt `-config`, který zpřístupňuje konfigurační soubor a objekt `-core`, kterým lze zpřístupnit vlastní jádro dbMan core a provádět tak (velmi nebezpečné) akce. O jeho využití se zmíním v závěru příspěvku.

Je vidět, že pro práci s daty používá dbMan celkem tři metody. V asociativním poli `%action` jsou uložena data potřebná pro vyřízení jedné akce, v objektu `$obj->{-mempool}` jsou uchovávána data do ukončení dbMana. Persistenci zajišťuje objekt `-config`.

Další funkce, které mohou být v rozšíření definovány, jsou např. `for_version()`, kterou lze ovlivnit, které dbMan core podporuje tento modul (kdyby došlo k zásadní změně API, což se zatím nestalo), `known_actions()`, které stanovuje, na jaké akce rozšíření reaguje a tak může zjednodušit a urychlit práci vlastní smyšky událostí a zjednodušit výstupy trasování (nepovinné). Metody `init()` a `done()` slouží pro provedení akcí při zavádění a odzavádění rozšíření (např. alokace promptu, definice údajů do datového skladu rozšíření apod.). Dvě další metody – `cmdhelp()` a `cmdcomplete()` jsou určeny pro definici nápovědných textů do centrální nápovědy a pro definici TAB kompletace konkrétního modulu (pro rozšíření realizující TAB kompletaci).

Objekt rozhraní umožňuje alokaci a dealokaci promptu (modifikace standardního `SQL:` na např. `LONG SQL:`) a funkce pro reálný výstup a práci s historií příkazů. Mempool nabízí funkce pro ukládání a získávání hodnot a plnění či rušení tzv. registrů (což jsou vektory údajů, např. seznam možných výstupních formátů).

6 Další možnosti modularity

Mimo uváděné definice rozšíření, můžeme přetěžovat interface objekty (pro definice nových rozhraní) – zde se očekává vyvinutí dalšího rozhraní pro práci s knihovnami `slang`/`ncurses` a rozhraní pro `qtgui` a `gtkgui`. Chci také dokončit stávající `tkgui`.

Existuje i možnost zasílání patchů na vlastní jádro dbMana přímo mne (a já rozhodnu, zda se budou do dbMana aplikovat), přidávání Vašich rozšíření či rozhraní do základní distribuce – nebo si můžete na CPANu publikovat vlastní rozšíření, přidat na `freshmeat.net` další vývojovou větev apod.

Perličkou na závěr může být fakt, že rozšíření implementují vše, tedy i vlastní zavádění a odzavádění jiných rozšíření (odzavedení rozšíření `Extensions` tedy může mít fatální následky).

7 Závěr

Dnes máme na univerzitě celé oddělení datamanagementu, které dbMana využívá. Já sám již datamanažerem nejsem (neb velikost informačního systému si vyžádala vznik velkého oddělení, které potřebuje „sekretářku“ shánějící peníze a podepisující faktury `:-)`), ale nadále rád věnuji dlouhé noci doplňováním nových rozšíření do dbMana podle požadavků našich datamanažerů i uživatelů z celého světa.

Budu rád, když využijete dbMana nebo myšlenky, které byly použity při jeho tvorbě, ve svých vlastních projektech. Ostatně o tom je myšlenka open-source. dbMan je distribuován pod Perlovou licencí, což Vám umožňuje volbu mezi GNU GPL licencí nebo Artistic licencí.