Masarykova univerzita
Institut biostatistiky a analýz
Kamenice 126/3
625 00 Brno

tel.: +420 549 49 4321
fax: +420 549 49 2855
e-mail: iba@iba.muni.cz
www.iba.muni.cz

pg_hist verze 1.0 – rozšíření pro PostgreSQL

technická a uživatelská dokumentace

Vypracoval: Kamil Morong, IBA MU, 15. 2. 2013

Masarykova univerzita
Institut biostatistiky a analýz
Kamenice 126/3
625 00 Brno

tel.: +420 549 49 4321
fax: +420 549 49 2855
e-mail: iba@iba.muni.cz
www.iba.muni.cz

Úvod

Podstatou rozšíření (extension) pro PostgreSQL je zapouzdření funkcí do snadno instalovatelného modulu. Rozšíření pg_hist je vyvinuto pro databázový systém PostgreSQL od verze 9.1. Jeho účelem je archivace historických záznamů v tabulkách databázového schématu včetně přidružených informací o původci a času změn. Historické záznamy je možné následně využít při auditování nebo obnově dat.

Struktura

Modul rozšíření sestává ze dvou textových souborů

1. *pg_hist.control* – metadatový soubor rozšíření pro PostgreSQL
2. *pg_hist--1.0.sql* – soubor s implementací funkcí

Všechny funkce jsou napsány v jazyku plpgsql.

Instalace

Soubory rozšíření je potřeba nakopírovat do adresáře extension v hierarchii PostgreSQL. Ve Windows to je např. *C:\Program Files\PostgreSQL\9.1\share\extension* a na Linuxu to může být */usr/share/postgresql/9.1/extension* (závisí na distribuci).

Instalaci do výchozího schématu databáze (nejčastěji public) provedeme spuštěním SQL příkazu CREATE EXTENSION pg_hist

Odstranění rozšíření provedeme spuštěním SQL příkazu DROP EXTENSION pg_hist, čímž dojde jen k odstranění z dané databáze, avšak soubory zůstávají na disku a lze tedy instalaci/deinstalaci provádět opakovaně v kterékoliv databázi.
Pro úspěšnou deinstalaci je nutné, aby žádný databázový objekt nebyl závislý na rozšíření – např. tabulky nesmí mít nastaveny triggery z rozšíření.

Princip

K uchování historických záznamů v tabulce dochází pomocí aplikace triggerů pro příkazy INSERT, UPDATE a DELETE. Tabulka, která tyto triggery používá, musí splňovat dvě podmíky:

- musí mít definován primární nebo unikátní klíč (i složený)
- musí obsahovat metadatové atributy (viz dále), tato podmínka je splněna při využití funkce, která na tabulku aplikuje sledování historie.

Při vložení nového záznamu příkazem INSERT dojde díky triggeru k automatickému vyplnění metadatového atributu ***dateCreated***. Uživatel musí při vkládání záznamu (kromě dat vyplývajících

Masarykova univerzita
Institut biostatistiky a analýz
Kamenice 126/3
625 00 Brno

tel.: +420 549 49 4321
fax: +420 549 49 2855
e-mail: iba@iba.muni.cz
www.iba.muni.cz

z definice tabulky) vyplnit metadatový atribut ***userCreated*** (integer – id uživatele). Ostatní metadatové atributy budou při vkládání ignorovány.

Při editaci záznamu příkazem UPDATE je naopak nutné vyplnit metadatový atribut ***userChanged*** (ostatní budou ignorovány), jinak nedojde k úspěšnému vykonání SQL příkazu.
Příkazem UPDATE nelze změnit žádný z metadatových atributů kromě ***userChanged***.

K odstranění záznamu z tabulky slouží funkce pgh_delete, jejíž syntaxe bude popsána následně. Důrazně doporučujeme **NEPOUŽÍVAT příkaz DELETE**, protože u něj nelze specifikovat uživatele provádějícího odstranění dat a tato poměrně zásadní operace by tak vykazovala nesprávné metadatové údaje.

Při editaci nebo smazání se původní záznam uloží do stínové tabulky. Název stínové tabulky je odvozen od názvu původní tabulky s přidaným sufixem *_hist* a nachází ve schématu, jehož název je rovněž odvozen od původního schématu s přidaným sufixem *_hist*. Např. tabulka *product* ve schématu *public* bude mít stínovou kopii *product_hist* ve schématu *public_hist*.
Vytvoření stínové tabulky a schématu je plně automatické a dojde k němu při první změně záznamu v tabulce.
Stínová tabulka kopíruje strukturu původní tabulky s tím rozdílem, že ignoruje NOT NULL deklarace, primární a unikátní klíč nahrazuje prostým indexem a přidává vlastní primární klíč ***id_hist*** a metadatový atribut ***deleted***.
Pokud dojde ke změně struktury původní tabulky, např. přidání sloupce, změně typu sloupce, je při prvním požadavku na zápis do stínové tabulky upravena také její struktura na základě těchto pravidel:

Originální tabulka	Stínová tabulka
odstranění sloupce	žádná změna
přidání sloupce	přidání sloupce
přejmenování sloupce	přidání sloupce s novým jménem
změna datového typu sloupce	přejmenování sloupce s připojením sufixu časového razítka a vytvoření nového sloupce s původním názvem a novým datovým typem

Porovnání struktury tabulek před a po aplikaci historie

Originální tabulka bez aplikace historie

public.product	
idProduct : integer	PRIMARY KEY
name : text	NOT NULL
startDate : date	NOT NULL
endDate : date	

Tabulka s aplikovanou historií

public.product	
idProduct : integer	PRIMARY KEY
name : text	NOT NULL
startDate : date	NOT NULL
endDate : date	
userCreated : integer	
userChanged : integer	
dateCreated : timestamp	
dateChanged : timestamp	

+ triggery pro INSERT, UPDATE a DELETE

Stínová tabulka

public_hist.product_hist	
id_hist : integer	PRIMARY KEY
deleted : boolean	
idProduct : integer	(INDEX)
name : text	
startDate : date	
endDate : date	
userCreated : integer	
userChanged : integer	
dateCreated : timestamp	
dateChanged : timestamp	

Přehled funkcí

Uživatelské funkce

V této kapitole je na příkladech uvedeno použití uživatelských funkcí z rozšíření.

Aplikaci historie na tabulku provedeme funkcí **pgh_add_table_history**. Funkce vytvoří v tabulce metadatové atributy a nastaví potřebné triggery. Podmínkou pro aplikaci historie je existence primárního nebo unikátního klíče v tabulce, v opačném případě je ohlášena výjimka.

Syntaxe:

pgh_add_table_history(*table_name* text [, *schema_name* text])
table_name – název tabulky
schema_name – název schématu, výchozí hodnota je *current_schema()* (nejčastěji public)

Příklad:

```
SELECT pgh_add_table_history('product');
```

Pokud se rozhodneme historii nepoužívat, potom může být odstraněna funkcí **pgh_remove_table_history**. Tato funkce provede odstranění triggerů a metadatových atributů, tzn., že **aktuální záznamy přijdou nenávratně o metadatové informace**. Stínová tabulka není nijak dotčena a je na uvážení uživatele, jak s ní naloží.

Syntaxe:

pgh_remove_table_history(*table_name* text [, *schema_name* text])
table_name – název tabulky
schema_name – název schématu, výchozí hodnota je *current_schema()* (nejčastěji public)

Příklad:

```
SELECT pgh_remove_table_history('product');
```

Masarykova univerzita
Institut biostatistiky a analýz
Kamenice 126/3
625 00 Brno

tel.: +420 549 49 4321
fax: +420 549 49 2855
e-mail: iba@iba.muni.cz
www.iba.muni.cz

Korektní odstranění záznamů z tabulky provedeme pomocí funkce **pgh_delete**. Tato funkce zabezpečí zapsání identifikace uživatele, který provedl odstranění, do záznamu ve stínové tabulce. Z tohoto důvodu není toto rozšíření vhodné tam, kde by byl pro manipulaci s daty v tabulkách použit některý ORM framework.

Syntaxe:

pgh_delete(*user_id* integer, *table_name* text [, *condition* text [, *using_clause* text]])
user_id – identifikace uživatele
table_name – název tabulky, může být včetně aliasu
condition – podmínka pro klauzuli WHERE
using_clause – specifikace dalších tabulek (mohou být včetně aliasů) při tzv. „join“ DELETE

Příklady:

SELECT pgh_delete(3, 'product');
Odstraní z tabulky *product* všechny záznamy a poznamená, že tak učinil uživatel s identifikátorem *3*.
Ekvivalent DELETE FROM product;

SELECT pgh_delete(3, 'product', 'endDate IS NOT NULL');
Odstraní z tabulky *product* záznamy s nenulovým *endDate*.
Ekvivalent DELETE FROM product WHERE endDate IS NOT NULL;

SELECT pgh_delete(3, 'product p', 'p.idProduct=pp.idProduct AND pp.idProducer=5', 'producerHasProduct AS pp')
Odstraní z tabulky *product* záznamy podle *idProduct*, které se nachází v tabulce *producer* a splňují podmínku *idProducer=5*.
Ekvivalent DELETE FROM product p WHERE p.idProduct=pp.idProduct AND pp.idProducer=5 USING producerHasProduct AS pp;

Interní funkce

Interní funkce začínají prefixem _ (podtržítko)a nejsou primárně určeny k samostatnému použití. Důrazně se nedoporučuje tyto funkce používat, protože to může vést k porušení funkčnosti sledování historie v databázových tabulkách

_pgh_2nd_name (*name* text)
Vrací název (text) s připojeným sufixem.

_pgh_add_column (*table_name* text, *schema_name* text, *column_name* text, *udt_name* text, *type_mod* integer, *array_dims* integer)
Do tabulky *table_name* ve schématu *schema_name* přidá sloupec dle specifikace *column_name, udt_name, type_mod, array_dims.*

_pgh_alter_column_stamp(*table_name* text, *schema_name* text, *column_name* text)
Přejmenuje sloupec *column_name* v tabulce *table_name* ze schématu *schema_name* tak, že k jeho názvu přidá textově formátované časové razítko (datum a čas)

_pgh_check_table_name(*full_table_name* text)
Parsuje zadaný název tabulky, který může být ve formátu *schema_name.table_name* včetně aliasu a vrací jeho unifikovaný tvar (text). Při chybném zápisu nebo neexistenci tabulky či schématu signalizuje výjimku.

_pgh_column_def(*column_name* text, *type_name* text, *type_mod* integer, *array_dims* integer)
Vrací definici sloupce (text) dle metadat ve tvaru vhodném pro příkaz ALTER TABLE ADD COLUMN.

_pgh_column_exists(*column_name* text, *table_name* text, *schema_name* text)
Ověřuje existenci sloupce *column_name* v tabulce *table_name* ze schématu *schema_name*, vrací boolean hodnotu.

_pgh_correct_atttypmod(*udt_name* text, *atttypmod* integer)
Vrací upravený modifikátor (integer) datového typu dle metadat PostgreSQL, například délka datového typu varchar.

_pgh_diff_tables(*table_name* text, *schema_name* text, *table_name2* text, *schema_name2* text)
Vrací sadu záznamů (record) s rozdílnými sloupci v tabulce *table_name* ze schématu *schema_name* oproti tabulce *table_name2* ve schématu *schema_name2*. Chybějící sloupce v tabulce *table_name* jsou ignorovány.

_pgh_drop_mandatory_columns(*table_name* text, *schema_name* text)
Z tabulky *table_name* ve schématu *schema_name* odstraní metadatové sloupce.

_pgh_drop_triggers(*table_name* text, *schema_name* text)
Z tabulky *table_name* ve schématu *schema_name* odstraní triggery zabezpečující funkčnost záznamu historie.

_pgh_ensure_2nd_schema(*schema_name* text)
Zajistí existenci stínového schématu k *schema_name*.

_pgh_ensure_2nd_table(*table_name* text, *schema_name* text)
Zajistí existenci stínové tabulky včetně aktualizace sloupců pro tabulku *table_name* ze schématu *schema_name*.

_pgh_ensure_mandatory_fields(*table_name* text, *schema_name* text)
Zajistí existenci potřebných metadatových atributů v tabulce *table_name* ze schématu *schema_name*.

_pgh_ensure_schema(*schema_name* text)
Zajistí vytvoření schématu *schema_name*.

_pgh_ensure_triggers(*table_name* text, *schema_name* text)
K tabulce *table_name* ve schématu *schema_name* připojí nutné triggery. Pokud již existují, jsou nejdříve odstraněny.

_pgh_full_table_name(*schema_name* text, *table_name* text)
Vrací název tabulky (text) včetně schématu ve formátu *schema_name.table_name*.

_pgh_get_session_var(*p_var_name* text)
Vrací hodnotu (text) „session“ proměnné *p_var_name*. Hodnota proměnné je uložena v dočasné tabulce, která existuje pouze pro danou databázovou session. Při neexistenci proměnné vrací NULL.

Masarykova univerzita
Institut biostatistiky a analýz
Kamenice 126/3
625 00 Brno

tel.: +420 549 49 4321
fax: +420 549 49 2855
e-mail: iba@iba.muni.cz
www.iba.muni.cz

_pgh_identity_columns(*table_name* text, *schema_name* text)
Vrací název (sadu názvů) sloupce(ů), které lze použít jako jednoznačný identifikátor v tabulce *table_name* ze schématu *schema_name*.

_pgh_identity_columns_str(*table_name* text, *schema_name* text)
Vrací čárkou oddělený seznam názvů sloupců (text), které lze použít jako jednoznačný identifikátor v tabulce *table_name* ze schématu *schema_name*.

_pgh_schema_exists(*schema_name* text)
Ověřuje existenci schématu *schema_name*, vrací boolean hodnotu.

_pgh_set_session_var(*p_var_name* text, *p_var_value* text)
Nastaví hodnotu „session" proměnné *p_var_name* na *p_var_value*. Hodnota proměnné se uloží do dočasné tabulky, která existuje pouze pro danou databázovou session. Při existenci proměnné je její hodnota přepsána.

_pgh_setup_2nd_table(*table_name* text, *schema_name* text, *table_name2* text, *schema_name2* text)
Zkopíruje strukturu tabulky *table_name* ze schématu *schema_name* do nově vytvořené *table_name2* ve schématu *schema_name2*. V nové tabulce vytvoří nový primární klíč a nastaví index na sloupci(ích) původního primárního klíče.

_pgh_suffix()
Vrací textovou „konstantu" sufixu pro stínovou tabulku či schéma (aktuálně *_hist*).

_pgh_table_columns(*table_name* text, *schema_name* text)
Vrací sadu názvů (text) sloupců tabulky *table_name* ze schématu *schema_name*.

_pgh_table_columns_str(*table_name* text, *schema_name* text)
Vrací čárkou oddělený seznam názvů sloupců (text) tabulky *table_name* ze schématu *schema_name*.

_pgh_table_defs(*table_name* text, *schema_name* text)
Vrací sadu definicí sloupů (record) tabulky *table_name* ze schématu *schema_name*.

_pgh_table_exists(*table_name* text, *schema_name* text)
Ověřuje existenci tabulky *table_name* ve schématu *schema_name*, vrací boolean hodnotu.

_pgh_table_has_identity(*table_name* text, *schema_name* text)
Ověřuje existenci sloupce, který slouží k identifikaci záznamů v tabulce *table_name* ze schématu *schema_name*. Vrací boolean hodnotu.

_pgh_table_has_mandatory_columns(*table_name* text, *schema_name* text)
Ověřuje existenci vyžadovaných sloupců v tabulce *table_name* ze schématu *schema_name*. Vrací boolean hodnotu.

_pgh_table_primary_key(*table_name* text, *schema_name* text)
Vrací název (sadu názvů) sloupce(ů), tvořící primární klíč v tabulce *table_name* ze schématu *schema_name*.

Masarykova univerzita
Institut biostatistiky a analýz
Kamenice 126/3
625 00 Brno

tel.: +420 549 49 4321
fax: +420 549 49 2855
e-mail: iba@iba.muni.cz
www.iba.muni.cz

_pgh_table_unique_key(*table_name* text, *schema_name* text)

Vrací název (sadu názvů) sloupce(ů), tvořící unikátní klíč v tabulce *table_name* ze schématu *schema_name*.

_pgh_tr_delete_row()

Funkce pro trigger volaný po vykonání příkazu DELETE.

_pgh_tr_insert_row()

Funkce pro trigger volaný před vykonáním příkazu INSERT.

_pgh_tr_update_row()

Funkce pro trigger volaný před vykonáním příkazu UPDATE.

_pgh_trigger_exists(*trigger_name* text, *table_name* text, *schema_name* text)

Ověřuje existenci triggeru *trigger_name* v tabulce *table_name* ze schématu *schema_name*, vrací boolean hodnotu.