Celem zadania jest opracowanie funkcjonalność magazynowania i filtorwania logów w bazie dla istniejącej usługi "logi".   
  
Założenia usługi:  
- aplikacje klienckie wysyłają logi na adres udp://logi.eclicto.pl:514 lub 5140 (dedykowany protokół bazujący na udp lub syslog)  
- usługa dostępna pod adresem logi.eclicto.pl przetrzymuje określoną pulę logów (max 50000 całość i 10000 dla jednego źródła) w pamięci  
- każdy log ma zestaw atrybutów, najważniejsze z nich to:  
    - source - źródło logu (nazwa aplikacji / usługi)  
    - client\_date - data klienta  
    - server\_date - data odebrania logu (serwera)  
    - kind - rodzaj logu (debug, info, warning, error, query, event itp)  
    - level - poziom logu (0..100)  
    - expire - czas życia logu (w dniach)  
    - uuid - identyfikator logu nadawany przez klienta (potrzebny w przypadku edycji, odwoływania się do logu)

- moduł powinien być przenośny w obrębie różnych baz – wszytko w schemacie „logs”, tak aby łatwo można było jednym zapytaniem dołączyć funkcjonalność do innego systemu i nie nastąpił konflikt nazw  
      
      
  
  
**Pozostałe atrybuty** (attr\_id, key, name, label, [required, multiple instances,] max\_length)  
    logger(1, "lgr", "logger", "Logger", 100),  
    clazz(2, "cls", "class", "Klasa", 300),  
    method(3, "mth", "method", "Metoda", 200),  
    level(4, "lvl", "level", "Poziom", 10),  
    levelName(5, "lvn", "level\_name", "Nawa poziomu", 30),  
    uid(6, "uid", "uid", "UID", 40),  
    kind(7, "knd", "kind", "Rodzaj", true, false, 30),  
    date(8, "dte", "date", "Data", true, false, 26),  
    nanoTimeStamp(80, "nts", "nanoTS", "Precyzyjny znacznik czasu", true, false, 10),  
    //----------  
    source(20, "src", "source", "Źródło", true, false, 100),  
    address(21, "adr", "address", "Adres", false, true, 100),  
    device(22, "dev", "device", "Urządzenie / UA", 200),  
    user(23, "usr", "user", "Użytkownik", 100),  
    tag(24, "tag", "tag", "Tag", false, true, 100),  
    value(25, "val", "value", "Wartość", 30000),  
    details(26, "det", "details", "Szczegóły", 30000),  
    comment(27, "com", "comment", "Komentarz", 200),  
    //-------  
    instance(30, "ist", "instance", "Instancja", 50),  
    session(31, "ses", "session", "Sesja", 50),  
    request(32, "req", "request", "Żądanie", 50),  
    version(33, "ver", "version", "Wersja", 20),  
    //--------  
    processId(40, "prc", "process\_id", "ID procesu", 10),  
    threadId(41, "thr", "thread\_id", "ID wątku", 10),  
    threadName(42, "thn", "thread\_name", "Nazwa wątku", 100),  
    threadPriority(43, "thp", "thread\_priority", "Priorytet wątku", 20),  
    //----  
    color(50, "fcl", "color", "Kolor czcionki", 20),  
    background(51, "bcl", "background", "Kolor tła", 20),  
    expireConsole(52, "cex", "expire\_console", "Czas wygasania w konsoli", 10),  
    expireDatabase(53, "dex", "expire\_database", "Czas wygasania w bazie danych", 10), // 0 - nie archiwizuj, < 0 lub null - automatycznie  
    url(54, "url", "url", "URL", false, true, 200),  
    //-----------  
    callStack(60, "cst", "call\_stack", "Stos metod", false, true, 300),  
    errorStack(61, "est", "error\_stack", "Stos błędów", false, true, 300),  
    attributes(62, "atr", "attributes", "Atrybuty", false, true, 1000),  
    data(63, "dta", "data", "Dane", false, true, 30000),  
    //---------  
    flags(70, "flg", "flags", "Flagi", false, true, 1000),  
    properties(71, "prp", "properties", "Właściwości", false, true, 100),  
    //-----------  
    fieldKey(101, "key", "key", "Klucz pola", 100),  
    progress(102, "prg", "progress", "Postęp", 100),  
    group(103, "grp", "group", "Grupa", 1000);  
  
  
  
  
**Struktura atrybutu :**  
    - attr\_type: typ atrybutu (lista powyżej)  
    - data\_type: typ danych (plain text, xml, json, lista, itp)  
    - values[]: wartości  
  
\* Niektóre atrybuty przyjmują wiele wartości, szczegóły w tabeli logs.d\_attr (poglądowo: może być nieaktualna)  
  
  
Zapis wartości logów:  
Treść przechowywana jest w tabeli logs."values":  
  values\_id integer     : klucz główny  
  hash integer,       : hasz wartości (połowa MD5) indeksowany, wykorzystywany do wyszukiwania wartości (tylko equals), hasz można wygenerować funkcją  logs.get\_hash(value text)   
  attr\_type smallint, : typ atrybutu (j.w.)  
  data\_type smallint, : typ danych (j.w.)  
  ref\_count integer, : ilość referencji  
  value text, : wartość tekstowa  
  
  
  
  
Logi dodawane są za pośrednictwem procedury logs.add(IN pck logs.lpacket) lub logs.add(IN p\_packets logs.lpacket[]).  
Po odebraniu paczki atrybutów danego logu, liczony jest hasz wartości, na jego podstawie sprawdzane jest istnienie wartości w słowniku, jeśli nie istnieje – jest dodawane. Możliwe jest też słownikowanie po stronie klienta. Klient w paczce przesyła oprócz wartości może przesłać również pole values\_id, wtedy nie ma konieczności przeszukiwania tabeli).

Atrybuty zapisywane są w tabeli logów jako tablica typów logs.filter\_packlet oraz w tabeli atrybutów (rozwiązanie nadmiarowe, docelowo trzeba wybrać lepszy wariant). Dwa atrybuty stanowią wyjątek (source, kind) i są zapisywane bezpośrednio w tabeli logów. Atrybuty te są wymagane i zostały przeniesione do tej tabeli ze względu na optymalizację.

**Cel zadania, filtrowanie logów, propozycja rozwiązania:**

Filtrowanie po stronie bazy będzie realizowane za pośrednictwem procedury logs.filter\_logs(p\_filters logs.filter\_packlet[]). Klient wywołuje procedurę podając niezbędne dane, w odpowiedzi dostaje zbiór identyfikatorów logów (odfiltrowanych).

**Założenia filtrów:**

- liczba wyników musi być ograniczona do danej wartości (ok. 10 000)

- zakres czasu musi być zadeklarowany (data minimalna)

- operatory filtrów to:

- równy (equals)

- rózny (not equals)

- zawiera (like)

- nie zawiera (not like)

- większy niż ( > )

- mniejszy niż ( < )

- typ operatora uzależniony jest od typu wartości (np. większy, mniejszy tylko dla daty i wartości liczbowych). O prawidłowe określenie typu dba klient

- wszystkie filtry przed przetworzeniem należy pogrupować

– typy atrybuty filtorowane jako iloczyn logiczny (AND),

- atrybuty uwzględniające (equals, like) danego typu jako suma (OR),

- atrybuty wykluczające (not equals, not like) danego typu jako iloczyn (AND)

- operatory wykluczające (not like, not equals) muszą mieć narzuconą minimalną długość (np. 3 znaki) (not like ‘%a%’ zwróci zbyt dużo wyników)

- należy zgrupować zakresy (mniejszy, większy)

przykłady

data\_serwera > „2014-12-01”, data\_sewera < „2014-12-03”, data\_serwera > „2014-12-10”, data\_serwera < „2014-12-20”, źródło = „Content”, źródło = „Tomcat”, Źródło != „Dev”, Adres = „127.0.0.1”, Adres = „localhost”, użytkownik like „Jan”, priorytet < 3, priorytet > 5, priorytet > 6

powinno dać rezultat:

((data\_serwera > „2014-12-01” AND data\_sewera < „2014-12-03”)

OR (data\_serwera > „2014-12-10”, data\_serwera < „2014-12-20”))

AND (źródło = „Content” OR źródło = „Tomcat”)

AND (Źródło != „Dev”)

AND (Adres = „127.0.0.1” OR Adres = „localhost”)

AND (użytkownik like „Jan”)

AND (priorytet > 3 AND priorytet < 5 AND priorytet < 6)

\* Zgrupowanie ostatniego warunku jest logicznie nie wykonalne.

filtrowanie powinno odbyć się w procedurze logs.filter\_logs(p\_filters logs.filter\_packet[])

﻿<filter>

<!-- Źródło = „Content” OR Źródło = „Tomcat” -->

<group attr\_type = "20">

<item opr = "1">Content</item>

<item opr = "1">Tomcat</item>

</group>

<!-- Źródło <> „Dev” -->

<group attr\_type = "20">

<item opr = "3">Dev</item>

</group>

<!-- Adres = „127.0.0.1” OR Adres = „localhost”) -->

<group attr\_type = "21">

<item opr = "1" id = "1224" >127.0.0.1</item>

<item opr = "1" id = "4354" >localhost</item>

</group>

<!-- użytkownik like „Jan”) -->

<group attr\_type = "23">

<item opr = "2">Jan</item>

</group>

<!-- ((data\_serwera > „2014-12-01” AND data\_sewera < „2014-12-03”)

OR (data\_serwera > „2014-12-10”, data\_serwera < „2014-12-20”))) -->

<group attr\_type = "-1">

<item opr = "2">

<group>

<item opr = "5">2014-12-01</item>

<item opr = "6">2014-12-03</item>

</group>

<group>

<item opr = "5">2014-12-10</item>

<item opr = "6">2014-12-20</item>

</group>

</item>

</group>

</filter>

TYPE logs.lattr AS (

attr\_type int,

data\_type int,

values text[],

values\_ids int[]

)

TYPE logs.filter\_packlet AS (

attr\_type int,

operator int,

value text

)

TYPE logs.lpacket AS (

uid UUID,

expire int,

client\_date timestamp without time zone,

server\_date timestamp without time zone,

level smallint,

attributes logs.lattr[]

)  
  
  
  
SELECT \* FROM **logs.add**(

('974358a0-7fd0-4901-8e40-23329ca5ef76',

null,

'2014-07-11 09:42:17.161',

'2014-07-11 09:42:11.543',

53,

ARRAY[

(20, 0, ARRAY[ (null, 'Logi') ]::logs.lvalue[] ),

(7, 0, ARRAY[ (null, 'request') ]::logs.lvalue[] ),

(22, 0, ARRAY[ (null, 'Mozilla/5.0 (Windows NT 6.1; WOW64)') ]::logs.lvalue[] ),

(23, 0, ARRAY[ (null, 'milosz.ziernik') ]::logs.lvalue[] ),

(21, 0, ARRAY[ (null, '10.1.0.254'), (null, '127.0.0.1') ]::logs.lvalue[] ),

(24, 0, ARRAY[ (null, 'Request') ]::logs.lvalue[] ),

(60, 0, ARRAY[ (null, 'com.logs.ServletLog.requestInfo (ServletLog.java:311)'),

(null, 'com.servlet.handlers.BPage.<init> (BPage.java:237)'),

(null, 'service.handlers.Page.<init> (Page.java:18)'),

(null, 'pages.PConsole.<init> (PConsole.java:26)'),

(null, 'sun.reflect.NativeConstructorAccessorImpl.newInstance0'),

(null, 'sun.reflect.NativeConstructorAccessorImpl.newInstance'),

(null, 'sun.reflect.DelegatingConstructorAccessorImpl.newInstance'),

(null, 'java.lang.Thread.run') ]::logs.lvalue[] ),

(2, 0, ARRAY[ (null, 'com.logs.ServletLog') ]::logs.lvalue[] ),

(3, 0, ARRAY[ (null, 'requestInfo (ServletLog.java:311)') ]::logs.lvalue[] ),

(1, 0, ARRAY[ (null, 'MLogger default v1.69 via udp://10.1.0.254:5140') ]::logs.lvalue[] ),

(40, 0, ARRAY[ (null, '9768') ]::logs.lvalue[] ),

(31, 0, ARRAY[ (null, '121') ]::logs.lvalue[] ),

(42, 0, ARRAY[ (null, 'Request, GET, /') ]::logs.lvalue[] ),

(43, 0, ARRAY[ (null, '5') ]::logs.lvalue[] ),

(32, 0, ARRAY[ (null, '56b95e7baa664b8ba36ba26f3b5035f6') ]::logs.lvalue[] ),

(31, 0, ARRAY[ (null, '334B2681F4DAA8780B1FBF4C1612E557') ]::logs.lvalue[] ),

(54, 0, ARRAY[ (null, 'http://milosz/logi/') ]::logs.lvalue[] ),

(33, 0, ARRAY[ (null, '1.5.1533') ]::logs.lvalue[] ),

(25, 0, ARRAY[ (null, 'GET, http://milosz/logi/') ]::logs.lvalue[] )

]::logs.lattr[])::logs.lpacket

);