

# Best practices to store IPv4/IPv6- addrs in (SQL) databases

## Swinog #21

Martin Blapp, <martin.blapp@nashire.com>

## Who's Nashire AG?

A startup company  
founded October 2010

We are E-Mail specialists

Located in Allschwil, Baselland

3 Founders

## Our main product line: Nashire Gate Suite

- Very flexible smtp cluster gateway for ISPs
- DDoS protection and functionality for SLAs
- Inbound/Outbound queues
- MS-Style SOAP 1.1 interface module
- Fast Log-search interface to find entries within billions of log lines
- Designed for millions of mails per day
- Multi-client, multi-routing capable
- Rule-Engine for client/domains/user
- SaaS ready
- Available as appliance / virtualized

and much more ...

## Nashire Gate Suite: Some screenshots

Gate 1.0, Logdaten aller Inbound-Verbindungen, Administrator@mx1.mail..., Class Client

Suche nach Zeilen wo Feld: Domain gleich demstac.ch Suche

Suchergebnisse der Suche auf 5 Feldern: Domain, Sender, Ser, Absteigend sortiert nach Feld Ser (pnpunkt)

MTA	Sender	Empfängerliste	Domain	Zeitpunkt	Typ	Klasse	Bounce	Grösse	Ident	Kommentar
IN			mx2.mail...	16-03-2010 23:59:26	ham	sm-mta-A	Nein	5144	<1e6b796ac0	-128.5 Spampunk ...
IN			mx2.mail...	16-03-2010 23:54:28	ham	sm-mta-A	Nein	2534	<2010031622	-1.21 Spampunk ...
IN			mx2.mail...	16-03-2010 23:53:18	spam	sm-mta-C	Nein	14056	report.1268777	6.552 Spampunk ...
IN			mx2.mail...	16-03-2010 23:52:01	ham	sm-mta-A	Nein	827	<1268779918	-0.9 Spampunk ...
IN			mx2.mail...	16-03-2010 23:50:12	ham	sm-mta-A	Nein	825	<1268779809	-0.9 Spampunk ...

Konfiguration - Mozilla Firefox

Komponenten

- Inbound Regel Gruppen
- Inbound Regel Aktionen
- Inbound Regel Kategorien
- Inbound Sender Klassen
- IP-Klassen
- Host-Klassen
- Datei-Typ Klassen
- Mime-Typ
- Regex Klassen
- Wiederholende Zeiträume
- Absgeschlossene Zeiträume

Sender Klassen

- Gruppe 1 (AWL pro Mandant)
- Gruppe 2 (AWL pro Domain)
- Gruppe 3 (AWL pro User)

Alle auswählen | Auswahl umkehren | Neue Gruppe erzeugen

Ausgewählte Gruppen löschen | Ausgewählte Gruppen kopieren

## Why this talk?

- A quick look at different projects showed:
  - Many implementations don't normalize databases
  - They use plain text fields to store numbers

## Storage Examples:

- config databases

Name	Routing
213.3.29.131	[213.3.29.131]
213.3.29.32	[213.3.29.32]
213.3.39.155	[213.3.39.155]
213.3.43.140	[213.3.43.140]

- firewall rules / kernel ip lists

```
> select net from kernel_dnswl
+-----+
| 109.104.64.0/24 |
| 109.106.51.0/24 |
| 109.74.192.0/24 |
| 109.74.197.0/24 |
| 112.213.92.0/24 |
+-----+
10 rows in set (0.02 sec)
```

- whitelists / blacklists / graylists

Oktett A	Oktett B	Oktett C	Oktett D	Status	Init. Verzögerung	Letzte Verzögerung	Erstellt am
95	29	213	88	GRAY	0	0	07-11-2010 12:43:38
189	70	106	217	GRAY	0	0	07-11-2010 12:43:38
201	43	177	199	GRAY	0	0	07-11-2010 12:43:38

- log indexes to logs (web / mail / access)

```
/app/var/log/maillog/2010-11-08]$ ls
00      03      06      09      12      15      18
01      04      07      10      13      16      19
02      05      08      11      14      17      20
```

## Why one needs to care about storage size these days?

- If you are ISP and have mail/access log indexes stored in DBs
- If you need small indexes and like to keep them in memory
- Blacklists with millions of entries (but only if they don't need to be unique)

## Some possible datatypes for IPv4 (32bit system)

Text:            varchar(15)                            '192.168.100.124'



Number:        1 x unsigned int                            3232261244

Number:        4 x unsigned smallint                    192            168            100            124

CIDR:            Special IPv4 and IPv6 datatype (including network postfixes)



## Size matters, some size calculations

IPv4, 1'000'000'000 entries:

Text: ~ 1400 MB

vs.

Number: 381 MB

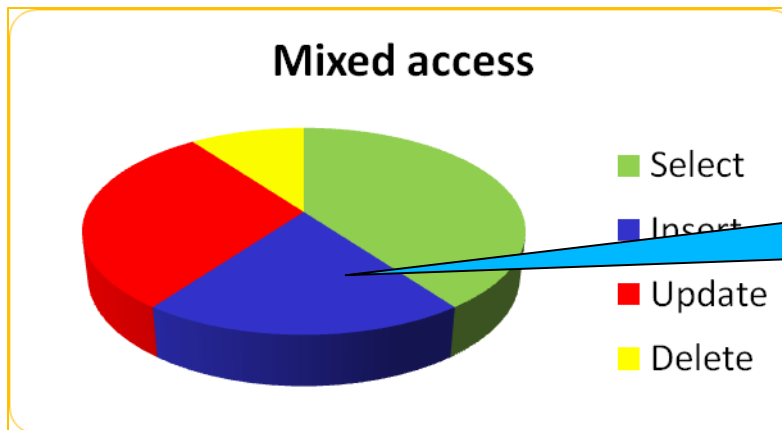
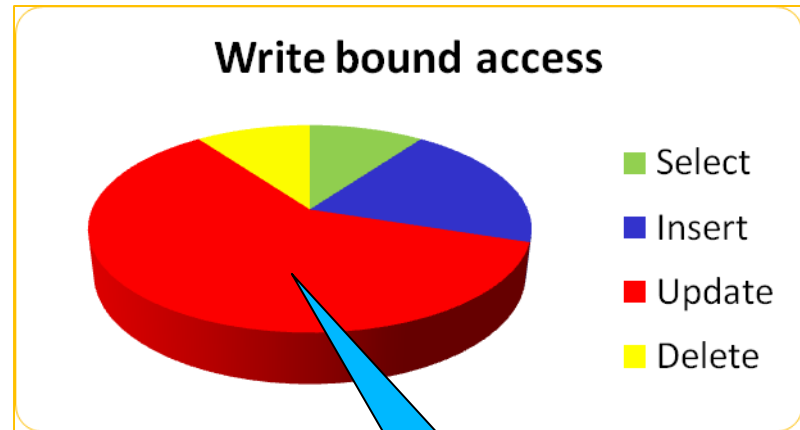
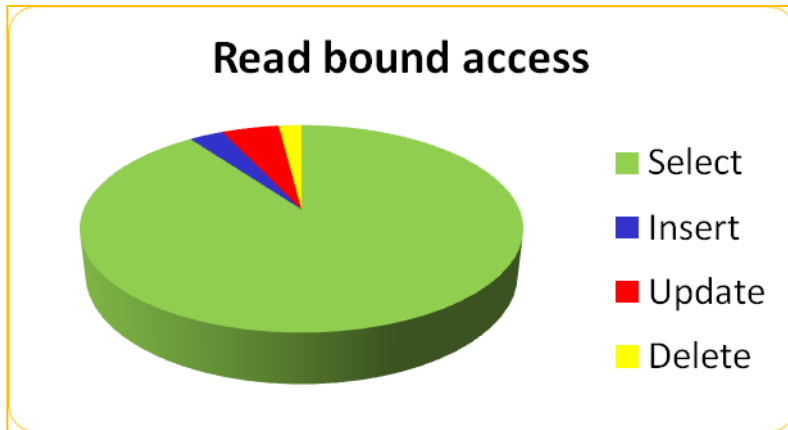
IPv6, 1'000'000'000 entries:

Text: ~ 3000 MB

vs.

Binary: 1525 MB

## More important: access patterns of your database



**Indexes slow down writes**

## Indexes are not free

- Indexes need disk space too
- They are only fast in memory
- Adding entries to the index / updating entries needs time
- We should really have a look at our access patterns
- Our IP address may not be the only indexed field
- We can use a partial index if the IP field isn't unique

## Indexes for IP-fields, unique or not ?

- Logs don't have unique IP-address fields
- Our IP address may be not be the only indexed field
- We can use a partial index if the IP field isn't unique

## To normalize or not to normalize, that's the question

- Depends on the use case
- You need to run statistics
- For intranet logs / statistics it definitely makes sense

## Some statistics about IPv4 usage:

IPv4 has four octets:    xxx.xxx.xxx.xxx    or    octa.octb.octc.octd

Example:    ISP maillog, 1000000 IP-Addrs:

octa            160 different values

octb            256 different values

octc            256 different values

octd            254 different values

## Smart and fast IPv4 index for mixed access

Solution: A partial index

- Partial Index on octb or octc      XXX.**XXX**.XXX.XXX    or    XXX.XXX.**XXX**.XXX

You can even add an additional tinyint field with the part index field octb or octc.

Int (11) unsigned

Tinyint (3) unsigned

## Some facts about IPv6 you all know:

IPv6 has 8 hexets:      XXXX:XXXX:XXXX:XXXX:XXXX:XXXX:XXXX:XXXX

Some remarks:

- The first three hexets will often stay the same
- Blacklists may only need the first four hexets (64bit)

## Smart and fast IPv6 indexes for mixed access

Some ideas:

- Partial Index on hextet four: XXXX:XXXX:XXXX:XXXX:XXXX:XXXX:XXXX:XXXX

You can even add an additional smallint field with the part index field:

varbinary (16) and smallint (5) unsigned

- Partial index for intranet use: XXXX:XXXX:XXXX:XXXX:XXXX:XXXX:XXXX:XXXX

## How to query/convert your data in SQL:

If you store numbers/binary IP-data you may need **functions** to convert:

IPv4:

inet\_aton      ( IP-Adress to Number )  
inet\_ntoa      ( Number to IP-Adress )

IPv6:

inet6\_pton    ( Presentation format to binary )  
inet6\_ntop    ( Binary format to presentation )

## Tricky SQL Functions for queries with many rows

```
SELECT ip FROM Tab_2a WHERE ip = inet6_pton(  
  '2001:fca9:b04:956:ce85:aa03:7de:740f');
```

1 row in set (**15.49 sec**)

**Called for  
each  
single row**

```
SET @t3 = inet6_pton(  
  '2001:fca9:b04:956:ce85:aa03:7de:740f');
```

```
SELECT ip FROM Tab_2a WHERE ip = @t3;
```

1 row in set (**2.80 sec**)

**Called  
only one  
time**

## Conclusions (not only for IP-addr storage)

- Design your database before you have slow query problems
- Think about how you or your customers access the data later
- Use smart partial indexes for IP-fields to make access really fast
- Think about the maximal size your DB will have in the future
- Avoid SQL-functions for static data if you have millions of rows

## We are looking for projects

- Email routing/filtering projects for ISPs, KMUs
- Unix (particularly FreeBSD and Linux) projects
- Mail and web- security audits and concepts
- Microsoft Backoffice: Exchange, Mail Security, Mail archiving solutions

**References:**

**FreeX 6/2010:** (german magazine)

Title: Optimal strategies to work with IP-address data in mysql

Pages: 48-56

Author: Martin Blapp

**Thank you**

Any questions?