TERENA TF-Cache
Small project 99-005
Extended Cache Statistics

Seafood - a log file analyser (Deliverable 3)

Jens-S. Vöckler 29. Oktober 1999

1 Database Implementation

The database schema was designed prefering more tables in the database in favour of less columns in each table. By spreading data over more tables, it is hoped that more intricate correlations are possible. All table names start with the `sf_*` prefix in order to avoid cluttering other tables already defined in the current namespace. Figure 1 shows the physical model of the set of tables which depend on the daily[1] interval I_1.

The physical model is based on a concrete database product, as can be seen with the types of the attributes. During development Oracle 7.3 was used. Other database products were also considered during the design of the tables and relationships. The schema is easily converted to different databases, if you follow the guidelines set below.

Figure 1 shows a circular arrangement of tables around the central table `st_stamp1`. All tables on the ring depend with part of their primary key on `I1ID`, which is the primary key of `st_stamp1`. In each outer ring table, the `I1ID` is tagged as foreign key.

The primary key fragment `I1ID` is an abstract number. Table `st_stamp1` uses an alternate key consisting of the combination of the cache host name and the start time stamp of interval I_1, the real key for the central table. The `I1ID` is a means of an *efficient* glue between the tables.

Due to the fact that there does not exist generic support for sequences, row identifiers, or triggers, the interval id generation is provided by the insertion script. If the database product allows, the dependency of the outer tables' interval id on the central id should be modelled using a foreign key or a `references` constraint on the outer tables' id column.

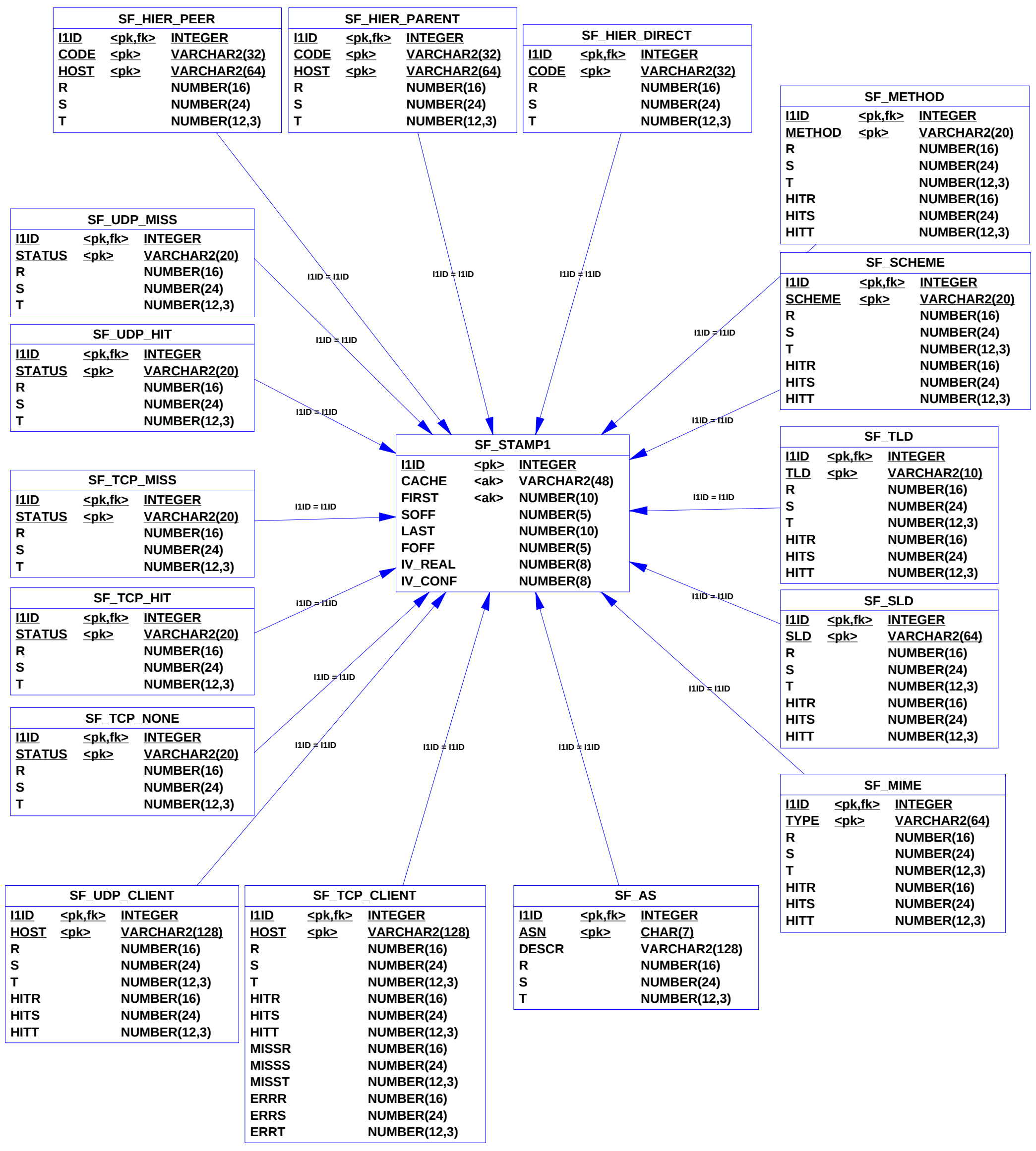
The central table `st_stamp1` contains the timestamp information about the daily interval for which seafood was run. It also uses the name of the cache it was run for, or the logical name of the cache group which seafood did sum up. Part of the table is also the offset to UTC in order to watch the daylight savings transitions. Finally, the interval for which the sum was run, and the configured interval are part of the table.

All tables on the ring contain the (R,S,T) triple with the following meaning:

- The R part is the request count.
- The S part is the volume sum in byte.
- The T part is the time spent in seconds, as reported in the log file. Hence, the UDP related data usually has no noticable transfer time, at least from the view of the log file.

1. See: http://www.cache.dfn.de/DFN-Cache/Development/Seafood/deliverable1.pdf

The DFN caching project is sponsored by the German Ministry of Education, Science, Research and Technology (BMBF) through the German Research Network Association (DFN Verein). The Trans-European Research and Education Network Association (TERENA) sponsors the "Extended Cache Statistics" project.



SF_HIER_PEER		
I1ID	<pk,fk>	INTEGER
CODE	<pk>	VARCHAR2(32)
HOST	<pk>	VARCHAR2(64)
R		NUMBER(16)
S		NUMBER(24)
T		NUMBER(12,3)

SF_HIER_PARENT		
I1ID	<pk,fk>	INTEGER
CODE	<pk>	VARCHAR2(32)
HOST	<pk>	VARCHAR2(64)
R		NUMBER(16)
S		NUMBER(24)
T		NUMBER(12,3)

SF_HIER_DIRECT		
I1ID	<pk,fk>	INTEGER
CODE	<pk>	VARCHAR2(32)
R		NUMBER(16)
S		NUMBER(24)
T		NUMBER(12,3)

SF_METHOD		
I1ID	<pk,fk>	INTEGER
METHOD	<pk>	VARCHAR2(20)
R		NUMBER(16)
S		NUMBER(24)
T		NUMBER(12,3)
HITR		NUMBER(16)
HITS		NUMBER(24)
HITT		NUMBER(12,3)

SF_UDP_MISS		
I1ID	<pk,fk>	INTEGER
STATUS	<pk>	VARCHAR2(20)
R		NUMBER(16)
S		NUMBER(24)
T		NUMBER(12,3)

SF_SCHEME		
I1ID	<pk,fk>	INTEGER
SCHEME	<pk>	VARCHAR2(20)
R		NUMBER(16)
S		NUMBER(24)
T		NUMBER(12,3)
HITR		NUMBER(16)
HITS		NUMBER(24)
HITT		NUMBER(12,3)

SF_UDP_HIT		
I1ID	<pk,fk>	INTEGER
STATUS	<pk>	VARCHAR2(20)
R		NUMBER(16)
S		NUMBER(24)
T		NUMBER(12,3)

SF_STAMP1		
I1ID	<pk>	INTEGER
CACHE	<ak>	VARCHAR2(48)
FIRST	<ak>	NUMBER(10)
SOFF		NUMBER(5)
LAST		NUMBER(10)
FOFF		NUMBER(5)
IV_REAL		NUMBER(8)
IV_CONF		NUMBER(8)

SF_TLD		
I1ID	<pk,fk>	INTEGER
TLD	<pk>	VARCHAR2(10)
R		NUMBER(16)
S		NUMBER(24)
T		NUMBER(12,3)
HITR		NUMBER(16)
HITS		NUMBER(24)
HITT		NUMBER(12,3)

SF_TCP_MISS		
I1ID	<pk,fk>	INTEGER
STATUS	<pk>	VARCHAR2(20)
R		NUMBER(16)
S		NUMBER(24)
T		NUMBER(12,3)

SF_TCP_HIT		
I1ID	<pk,fk>	INTEGER
STATUS	<pk>	VARCHAR2(20)
R		NUMBER(16)
S		NUMBER(24)
T		NUMBER(12,3)

SF_SLD		
I1ID	<pk,fk>	INTEGER
SLD	<pk>	VARCHAR2(64)
R		NUMBER(16)
S		NUMBER(24)
T		NUMBER(12,3)
HITR		NUMBER(16)
HITS		NUMBER(24)
HITT		NUMBER(12,3)

SF_TCP_NONE		
I1ID	<pk,fk>	INTEGER
STATUS	<pk>	VARCHAR2(20)
R		NUMBER(16)
S		NUMBER(24)
T		NUMBER(12,3)

SF_MIME		
I1ID	<pk,fk>	INTEGER
TYPE	<pk>	VARCHAR2(64)
R		NUMBER(16)
S		NUMBER(24)
T		NUMBER(12,3)
HITR		NUMBER(16)
HITS		NUMBER(24)
HITT		NUMBER(12,3)

SF_UDP_CLIENT		
I1ID	<pk,fk>	INTEGER
HOST	<pk>	VARCHAR2(128)
R		NUMBER(16)
S		NUMBER(24)
T		NUMBER(12,3)
HITR		NUMBER(16)
HITS		NUMBER(24)
HITT		NUMBER(12,3)

SF_TCP_CLIENT		
I1ID	<pk,fk>	INTEGER
HOST	<pk>	VARCHAR2(128)
R		NUMBER(16)
S		NUMBER(24)
T		NUMBER(12,3)
HITR		NUMBER(16)
HITS		NUMBER(24)
HITT		NUMBER(12,3)
MISSR		NUMBER(16)
MISSS		NUMBER(24)
MISST		NUMBER(12,3)
ERRR		NUMBER(16)
ERRS		NUMBER(24)
ERRT		NUMBER(12,3)

SF_AS		
I1ID	<pk,fk>	INTEGER
ASN	<pk>	CHAR(7)
DESCR		VARCHAR2(128)
R		NUMBER(16)
S		NUMBER(24)
T		NUMBER(12,3)

Figure 1: Physical model of tables depending on the daily interval.

If one is using a different database product, e.g. one which supports native arithmetic types, R and S should be assigned an eight byte integers, and T an eight byte float. Even though seafood only uses a four byte integer for its internal request counters, the database may view a much larger interval. For this reason, the sum of a year's worth of requests might overflow some internal database registers, and some DB products report the wrong results instead of an error.

Some tables contain the related (HITR,HITS, HITT) triples. These triples refer to the number HITs[1] perceived by seafood. Please note that a HIT triple always contains values less or equal to the plain (R,S,T) triple in the same table and row. Furthermore, the `sf_tcp_-`

1. For a definition of what is considered a HIT by seafood: http://www.cache.dfn.de/DFN-Cache/Development/Seafood/deliverable2-3.pdf, section 5.1.

`client` table contains a (MISSR,MISSS,MISST) triple and an (ERRR,ERRS,ERRT) for errors and those non-HITs with a hierarchy code of NONE.

Starting on the left side of the ring, the UDP and TCP counts are collected in five tables `sf_udp_*` and `sf_tcp_*`. Besides the interval id as a glue, the primary key contains also the status code from the fourth column of the `access.log` file, e.g. `TCP_MEM_HIT`.

The top of figure 1 shows the tables dealing with the server side traffic of the squid. The hierarchy code is part of the primary key. For a sibling and a parent, the host contacted is also part of the key. When going directly to the origin site, the destination host is neglected, even though seafood knows about them internally.

On the right side of the ring, the request method, the scheme, the top-level, the 2nd-level domain, and the mime types are counted. All five tables contain the HIT count besides the plain count. Part of the primary key of the method table, the scheme table, the top-level domains and the mime types are only those items which were configured in `seafood.conf` by the time seafood was run. Furthermore, the number of entries in the domain tables for a particular cache host and timestamp are limited to the top N domains[1] as configured with seafood.

The direct hosts can be combined into networks and autonomous systems, if a whois service is available, and seafood was thus configured. The `sf_as` table contains the destination AS number of origin servers which were visited directly. Additionally, the description for the AS as seen during the seafood run is part of the database.

Finally, the client tables `sf_udp_client` and `sf_tcp_client` deal with the client side of the cache. Part of the primary key is the client host address. The format of the client host, symbolic or numeric, is taken as-is from the seafood provided output. Note that the number of clients is limited to the top N clients[2] as configured in seafood. Especially for the TCP clients, is seems feasible to provide a cut-off threshold of some percentage of the requests in order to weed out clients whose access is denied. Please note that such an alternative is not part of the current project.

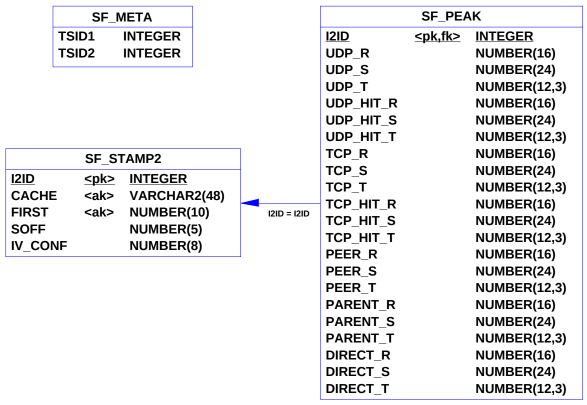


Figure 2: Remaining tables.

Figure 2 shows the `sf_stamp2` table, which deals with the hourly interval I_2. Only the start time of the interval and the UTC offset is part of the table. Again the start timestamp and the

1. Currently, only a ranking by requests is provided.
2. For now, only a ranking by requests is provided.

cache hostname are alternate keys for the interval id. Please note that this time interval id `I2ID` is used.

Only the performance statistics table `sf_peak` depends on the second stamp table. For each hourly time stamp, it collects the UDP and TCP data in HITs and MISSes, as well as the hierarchy data. Not part of the table is the count for objects with the hierarchy code NONE.

Finally, the `sf_meta` table records the highest number the interval IDs currently have. The insertion script makes use of these numbers, increases them, and generates appropriate SQL statements using the new interval IDs.

2 Usage

This section deals in an examplary way with the plumbing of data from seafood into a database. It is a kind of user guideline, but at the stage of deliverable D3, the user cannot do much apart from filling the database with data.

The examples shown in this section are based on Oracle 7.3, but due to the fact that a scripting language like Perl is used, you should be able to easily modify it to suit your needs.

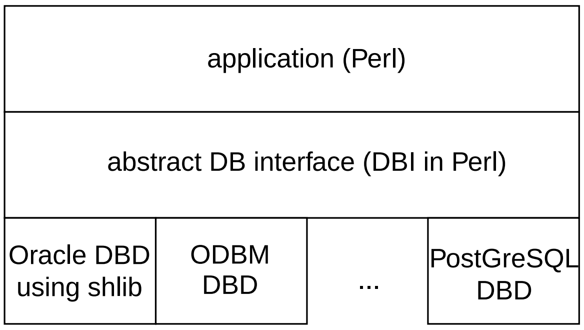


Figure 3: Perl DB Communication Interface.

Figure 3 shows the communication stack of Perl's database interface. The application calls generic functions of the abstract DBI interface. The interface itself, using a connect string, maps the calls to the underlying concrete database driver (DBD) which communicates with the database.

2.1 Installation

In order to use seafood, you will need to create the tables in your database, once only. Also, the meta information table sf_meta needs to be filled with the default pair (0,0). For Oracle, a Perl script is part of seafood. Start your SQL interfacing program, and run the script. For Oracle, you would need to start `sqlplus` and use the at (@) char to run an external file, e.g.:

```
voeckler@blau:~ $ sqlplus
SQL*Plus: Release 3.3.3.0.0 - Production on Thu Oct 28 11:58:38 1999
[... username/password ...]
SQL> @create-ora.sql
Table created.
1 row created.
Table created.
[... more "Table created" messages ...]
Table altered.
[... more "Table altered" messages ...]
SQL>
```

You must modify the `create-ora.sql` script before starting it to suit the concrete database product you are using. For instance, PostGreSQL does not know about foreign keys, but is able to use a `references` column constraint during table construction.

After you created the table, they will be visible in the DBMS data dictionary. For Oracle, you can use the following query to look up the tables for the current user:

```
SQL> select table_name from sys.user_tables;
```

```
TABLE_NAME
------------------------------
SF_AS
SF_HIER_DIRECT
SF_HIER_PARENT
SF_HIER_PEER
SF_META
SF_METHOD
SF_MIME
SF_PEAK
SF_SCHEME
SF_SLD
SF_STAMP1
SF_STAMP2
SF_TCP_CLIENT
SF_TCP_HIT
SF_TCP_MISS
SF_TCP_NONE
SF_TLD
SF_UDP_CLIENT
SF_UDP_HIT
SF_UDP_MISS

20 rows selected.
```

2.2 Filling the database

After you created the tables in your database, you are ready to fill in data. The seafood program has been extended since deliverable D2. Usually, seafood will show its results in textual tabular form on stdout, but if the new -D flag is used, the results will be put into an alternative output file meant to be parsed by the database insertion script.

In order to create data to be filled into your database, you have to run seafood with the -D flag first:

```
$ ./seafood -f ../seafood.conf -D hamburg.data ../hamburg.log
[...]
```

The example above will use the alternative configuration file as specified, parse the log file hamburg.log and produce the results in hamburg.data. The result file is pure textual, and meant to be parsed by the perlish insertion script.

The example script is based on Oracle 7.3. The script needs some environment variables and script variables to be set in order to work correctly. Refer to table 1 and table 2 for more insight, which variables need to be set.

The scripts all rely on Perl 5.005 or above to be installed, and uses the DBI extensions as well as the DBD::Oracle extension. Refer to a CPAN mirror[1] near you for the latest modules.

The script also expects the generic module Tables.pm, part of the seafood suite. The Tables module consists of two items, a list `@tables` with all name of the tables in your database, and a generic insertion method. Later, this method will be rewritten to also provide update capabilities, but for now, only insertion into the database is possible.

1. http://www.cpan.org/

Variable	Remarks
$dbhost	Host name of your database server.
$dbname	Name of the DBD driver module, e.g. ’Oracle’.
$dbuser	Name of user to connect to database
$dbpass	Password of database user.
$dborg	Database connection name.

Table 1: Script variables to be set for Oracle and Perl.

Variable	Remarks
ORACLE_SID	Database connection name for local machines.
ORACLE_USERID	User and (possibly) password, slash separated.
ORACLE_BASE	Base directory of your Oracle installation.
ORACLE_TERM	Name of the terminal emulation, e.g. "xsun5".
ORACLE_HOME	Base directory of the Oracle version installation.
TWO_TASK	Database connection name for remote machines.
NLS_LANG	Character set to be used (American_America.we8iso8859p1").
PATH	Should contain ${ORACLE_HOME}/bin.

Table 2: Environment variable to be set for Oracle.

The insertion script starts out by loading the concrete database driver and connecting to the database. In the next step, it tries to verify that all necessary tables are to be found in the data dictionary of your system. Finally, it reads the seafood output and stores the data into the database. If an error is detect, the insertion will be rolled back and the script aborted. Otherwise, success will be reported and the changes commited.

```
$ perl insert-ora.pl hamburg.data
# installing database driver
# trying to connect to DB
# trying to verify tables.
# found [SF_AS]
   [...]
# found [SF_UDP_MISS]
# reading input
# stamp1...
# tcp...
# udp...
   [...]
# as...
# dist...
SUCCESS!
```

At the moment, you are unable to insert the same file twice. A more accurate insertion should be possible, e.g. checking the first stamp and cache host from the time stamp tables. If detected, updates should be used, either adding or overwriting existing data. But at the moment, only insertion of new rows can be used.

2.3 Deinstallation

In case you want to remove all tables from your database, you just need to drop the appropriate tables. Some products like PostGreSQL are not capable of cascading dependent drops, so you might need to expand the uninstall SQL script in order to drop automatically constructed indices.

Beware, running the uninstall script will remove all seafood related data from your database! You could lose years worth of data.

```
voeckler@blau:~ $ sqlplus
[... username/password ...]
SQL> @destroy-ora.sql
Table dropped.
[... more "Table dropped" messages ...]
SQL> commit;
```

2.4 Summary

Table 3 shows the programs and scripts described so far. Please not that the Oracle dialect is used in all the examples. Scripts for other database dialects were also supplied, but not tested.

Program	Meaning	Language
seafood	log file analyzer.	C++, binary
create-ora.sql	Database table creation script.	SQL
destroy-ora.sql	Database destruction script.	SQL
insert-ora.pl	Database value insertion script, which takes the results from seafood and puts them into the database.	Perl
Tables.pm	Helper module for the insertion script.	Perl

Table 3: Scripts and programs supplied.

3 Thoughts on the current implementation

The current database insertion is more or less a case study using Oracle. It is expected to provide an improved version by the time the web user interface is implemented, especially using the insights from the querying point of view.

- The daily interval I_1 can be configured, but is at the moment utterly unused. It is assumed that the log file(s) fed into seafood are just about matching the configured interval. If fed with a larger interval, seafood will *not* split the results into several files.
- The database feeder work by using Perl DBC. A generic approach can only use Perl or Java as programming languages dealing with the task. The use of embedded SQL in C++ is not feasible, as some vendors do not supply pre processors, other vendors are only capable of parsing ANSI-C and the overall product is very tightly coupled to the database product. Still, it is assumed that using Java should yield an even more portable result though this is *not* part of the current project.
- Only Oracle examples are provided. For the sake of general acceptance, some other database vendors should be included, too. Informix, ANSI-SQL and ODBM-SQL scripts are also supplied, but not tested. PostGreSQL seems feasible, too. MS Access *cannot* be used due to its limited data range - it does not support 64 bit integers or some comparable data type.
- Not all tags provided by the seafood database output are really put into the database. At the moment, the output for the request distribution is excluded.
- Currently, only a top N implementation is used for five tables. Alternatively, other approaches like a chosen set or a percentual threshold are noted, but not part of the project.

3.1 The future

Originally the intention was to support any database, as long as the database is able to understand ANSI SQL. In the meantime, without deeper knowledge of what is defined in ANSI about SQL, and what is not, it is deemed feasible to target industrial grade databases like Informix, Oracle, Adabas and the like. Those database either already provide, or will in the near future provide special capabilities for handling data warehouses.

The project proposal refers to correlating data[1], trends not immediately obvious from the basic data. The goals of "data mining" are to detect, interpret and predict qualitative and quantitative patterns in the data[2]. The proposed "extended cache statistics" will cerntainly not do all this, but it is believed that storing well-selected basic data is the first step into the right direction, and it is hope that seafood, when finished, is able to provide at least some glimpses of what is possible with the data.

1. See: http://www.cache.dfn.de/DFN-Cache/Development/Seafood/.
2. IEEE computer magazine, Aug ’99.