



Pandore – Database developer guide

This document is a developer guide for the Pandore project database

Clément LE GRUIEC

Salma CHAHMI

Nathan OLBORSKI

Hugo HOUILLON



IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Table des matières

1.	General informations	4
1.1.	Server.....	4
1.2.	Database architecture	4
2.	Versions	5
2.1.	Version 1.....	5
2.2.	Version 2.....	5
3.	Table Service.....	6
3.1.	Description	6
3.2.	Stored procedures.....	7
3.2.1.	CreateService.....	7
3.2.2.	ReadAllServices.....	8
3.2.3.	ReadServiceByID.....	9
3.2.4.	ReadServiceByName.....	10
3.2.5.	UpdateService	11
3.2.6.	DeleteServiceByID	12
3.2.7.	DeleteServiceByName	13
4.	Table Server	14
4.1.	Description	14
4.2.	Stored procedures	15
4.2.1.	CreateServer	15
4.2.2.	ReadAllServers.....	16
4.2.3.	ReadServersByServiceID	17
4.2.4.	ReadServerByID	18
4.2.5.	ReadServerByAddress	19
4.2.6.	UpdateServer.....	20
4.2.7.	DeleteServerByID	21
4.2.8.	DeleteServerByAddress	22
4.2.9.	DeleteServerByServiceID.....	23
5.	Table DNS	24
5.1.	Description	24
5.2.	Stored procedures	25
5.2.1.	CreateDNS	25
5.2.2.	ReadAllDNS.....	26
5.2.3.	ReadDNSByServiceID	27
5.2.4.	ReadDNSByID.....	28

5.2.5.	ReadServerByAddress	29
5.2.6.	UpdateDNS	30
5.2.7.	DeleteDNSByID	31
5.2.8.	DeleteDNSByValue	32
5.2.9.	DeleteDNSByServiceID	33
6.	Table Capture	34
6.1.	Description	34
6.2.	Stored procedures	35
6.2.1.	CreateCapture	35
6.2.2.	ReadAllCaptures	36
6.2.3.	ReadCaptureByID	37
6.2.4.	UpdateCapture	38
6.2.5.	DeleteCaptureByID	39
7.	Table Capture_Request	40
7.1.	Description	40
7.2.	Stored procedures	41
7.2.1.	CreateRequest	41
7.2.2.	CreateRequestString.....	42
7.2.3.	ReadAllRequests	43
7.2.4.	ReadRequestsByCaptureID.....	44
7.2.5.	ReadRequestByID	45
7.2.6.	UpdateRequest	46
7.2.7.	DeleteRequestByID.....	47
7.2.8.	DeleteRequestByCaptureID.....	48

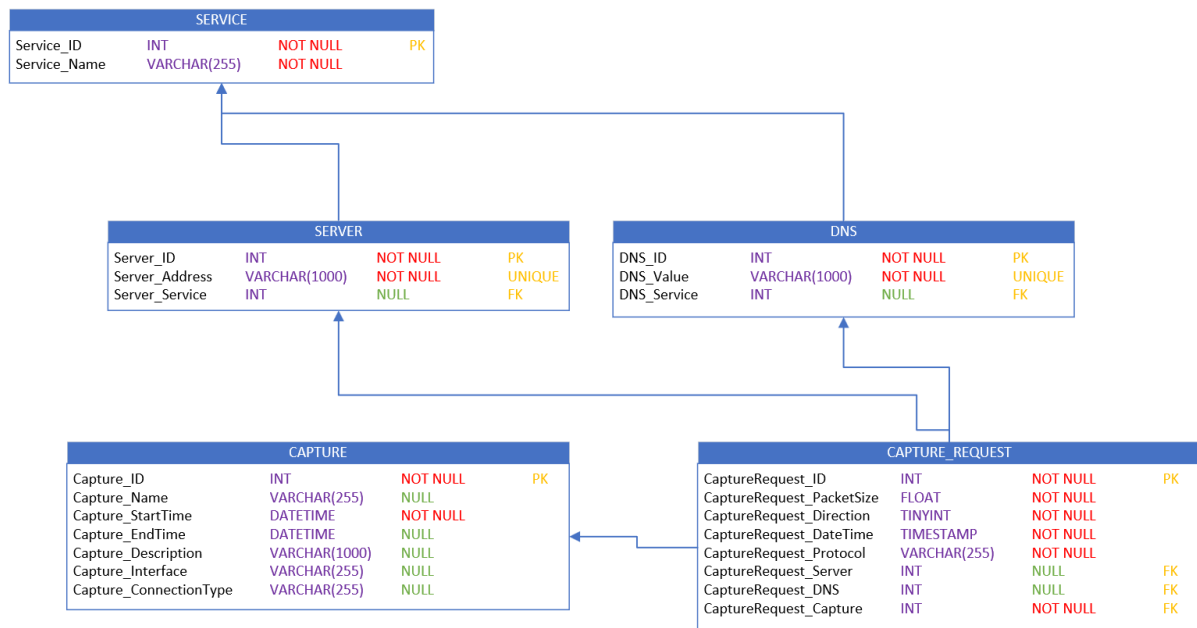
1. General informations

1.1. Server

The database server used for the Pandore project is **MariaDB Server 10.6.4**

1.2. Database architecture

The diagram below is a class diagram of the database architecture:



2. Versions

2.1. Version 1

- First database architecture implementation

2.2. Version 2

- Fields *CaptureRequest_DNS* and *CaptureRequest_Server* now allow *NULL* values
- Stored procedure *CreateRequestString* edited. It allows now to call it with *NULL* values for *DNS* or *Server* fields
- Stored procedure *ReadAllRequests* edited. New procedure parameter *Details*. Allows now to get requests with DNS and Server details
- Stored procedure *ReadRequestsByCaptureID* edited. New procedure parameter *Details*. Allows now to get requests with DNS and Server details
- Stored procedure *ReadRequestByID* edited. New procedure parameter *Details*. Allows now to get requests with DNS and Server details

3. Table Service

3.1. Description

This table is used to identify the name of different services (ex : Facebook, Youtube, ...).

It consists of 2 attributes :

Name	Type	Allow NULL	Constraint	Description
Service_ID	INT		PK	Service ID
Service_Name	VARCHAR(255)		Unique	Service Name

3.2. Stored procedures

3.2.1. CreateService

3.2.1.1. Description

This procedure allows to create a new service into the database

3.2.1.2. Prototype

```
CreateService (IN Name VARCHAR(255))
```

3.2.1.3. Parameters

Name	Type	Allow NULL	Description
Name	VARCHAR(255)		Service name

3.2.1.4. Example

```
CALL CreateService("Facebook");
```

3.2.2. ReadAllServices

3.2.2.1. Description

This procedure allows to list all services in the database

3.2.2.2. Prototype

ReadAllServices()

3.2.2.3. Returns

Service list ordered alphabetically

```
MariaDB [pandore]> CALL ReadAllServices();
```

Service_ID	Service_Name
1	Facebook
3	Google
2	Youtube

3.2.2.4. Example

CALL ReadAllServices();

3.2.3. ReadServiceByID

3.2.3.1. Description

This procedure allows to find a service in the database by its ID

3.2.3.2. Prototype

ReadServiceByID(IN ID INT)

3.2.3.3. Parameters

Name	Type	Allow NULL	Description
ID	INT		Service ID to find

3.2.3.4. Returns

The service found with the given ID

```
MariaDB [pandore]> CALL ReadServiceByID(1);
+-----+-----+
| Service_ID | Service_Name |
+-----+-----+
|          1 | Facebook     |
+-----+-----+
```

3.2.3.5. Example

CALL ReadServiceByID(1);

3.2.4. ReadServiceByName

3.2.4.1. Description

This procedure allows to find a service by its name

3.2.4.2. Prototype

```
ReadServiceByName(IN Name VARCHAR(255))
```

3.2.4.3. Parameters

Name	Type	Allow NULL	Description
Name	VARCHAR(255)		Service name to find

3.2.4.4. Returns

Services found with the given service name

```
MariaDB [pandore]> CALL ReadServiceByName("Facebook");
+-----+-----+
| Service_ID | Service_Name |
+-----+-----+
|          1 | Facebook     |
+-----+-----+
```

3.2.4.5. Example

```
CALL ReadServiceByName("Facebook");
```

3.2.5. UpdateService

3.2.5.1. Description

This procedure allows to update an existing service

3.2.5.2. Prototype

```
UpdateService(IN ID INT, IN Name VARCHAR(255))
```

3.2.5.3. Parameters

Name	Type	Allow NULL	Description
ID	INT		Service ID to update
Name	VARCHAR(255)		New service name

3.2.5.4. Example

```
CALL UpdateService(1, "Google");
```

3.2.6. DeleteServiceByID

3.2.6.1. Description

This procedure allows to delete a service by its ID

3.2.6.2. Prototype

DeleteServiceByID (**IN** ID **INT**)

3.2.6.3. Parameters

Name	Type	Allow NULL	Description
ID	INT		ID of the service to delete

3.2.6.4. Example

CALL DeleteServiceByID(1);

3.2.7. DeleteServiceByName

3.2.7.1. Description

This procedure allows to delete a service by its name

3.2.7.2. Prototype

DeleteServiceByName (**IN** **Name** **VARCHAR**(255))

3.2.7.3. Parameters

Name	Type	Allow NULL	Description
Name	VARCHAR (255)		Name of the service to delete

3.2.7.4. Example

CALL DeleteServiceByName("Facebook");

4. Table Server

4.1. Description

This table is used to identify different server addresses and associate them to services

It consists of 3 attributes :

Name	Type	Allow NULL	Constraint	Description
Server_ID	INT		Primary Key	Service ID
Server_Address	VARCHAR(1000)		Unique	Service Name
Server_Service	INT		Foreign Key	ID of the associated service

4.2. Stored procedures

4.2.1. CreateServer

4.2.1.1. Description

This procedure allows to create a new server in the database

4.2.1.2. Prototype

```
CreateServer(IN Address VARCHAR(1000), IN Service INT)
```

4.2.1.3. Parameters

Name	Type	Allow NULL	Description
Address	VARCHAR(1000)		Address of the server
Service	INT		ID of the associated service

4.2.1.4. Example

```
CALL CreateServer("10.135.20.32", 1);
```

4.2.2. ReadAllServers

4.2.2.1. Description

This procedure allows to list all servers in the database

4.2.2.2. Prototype

ReadAllServers()

4.2.2.3. Returns

Server list ordered alphabetically

```
MariaDB [pandore]> CALL ReadAllServers();
```

Server_ID	Server_Address	Server_Service
2	10.135.20.31	1
3	10.135.20.37	1
4	125.15.02.171	2

4.2.2.4. Example

CALL ReadAllServices();

4.2.3. ReadServersByServiceID

4.2.3.1. Description

This procedure allows to list all servers for a given service

4.2.3.2. Prototype

ReadServersByServiceID(IN Service INT)

4.2.3.3. Parameters

Name	Type	Allow NULL	Description
Service	INT		ID of the service

4.2.3.4. Returns

List of servers associated to given service

```
MariaDB [pandore]> CALL ReadServersByServiceID(1);
+-----+-----+-----+
| Server_ID | Server_Address | Server_Service |
+-----+-----+-----+
|          2 | 10.135.20.31   |              1 |
|          3 | 10.135.20.37   |              1 |
+-----+-----+-----+
```

4.2.3.5. Example

CALL ReadServersByServiceID(1);

4.2.4. ReadServerByID

4.2.4.1. Description

This procedure allows to find a server by its ID

4.2.4.2. Prototype

ReadServerByID(IN ID INT)

4.2.4.3. Parameters

Name	Type	Allow NULL	Description
ID	INT		ID of the server

4.2.4.4. Returns

Server found with given ID

```
MariaDB [pandore]> CALL ReadServerByID(2);
+-----+-----+-----+
| Server_ID | Server_Address | Server_Service |
+-----+-----+-----+
|          2 | 10.135.20.31   |                1 |
+-----+-----+-----+
```

4.2.4.5. Example

CALL ReadServerByID(1);

4.2.5. ReadServerByAddress

4.2.5.1. Description

This procedure allows to find a server by its address

4.2.5.2. Prototype

```
ReadServerByAddress(IN Address VARCHAR(1000))
```

4.2.5.3. Parameters

Name	Type	Allow NULL	Description
Address	VARCHAR(1000)		Address of the server

4.2.5.4. Returns

Server found with given address

```
MariaDB [pandore]> CALL ReadServerByAddress("10.135.20.31");
+-----+-----+-----+
| Server_ID | Server_Address | Server_Service |
+-----+-----+-----+
|          2 | 10.135.20.31  |              1 |
+-----+-----+-----+
```

4.2.5.5. Example

```
CALL ReadServerByAddress("10.135.20.31");
```

4.2.6. UpdateServer

4.2.6.1. Description

This procedure allows to update a server

4.2.6.2. Prototype

```
UpdateServer(IN ID INT, IN Address VARCHAR(1000), IN Service INT)
```

4.2.6.3. Parameters

Name	Type	Allow NULL	Description
ID	INT		ID of the server to update
Address	VARCHAR(1000)		New server address
Service	INT		New service ID

4.2.6.4. Example

```
CALL UpdateServer(1, "147.112.200.11", 2);
```

4.2.7. DeleteServerByID

4.2.7.1. Description

This procedure allows to delete a server by its ID

4.2.7.2. Prototype

DeleteServerByID(IN ID INT)

4.2.7.3. Parameters

Name	Type	Allow NULL	Description
ID	INT		ID of the server to delete

4.2.7.4. Example

CALL DeleteServerByID(1);

4.2.8. DeleteServerByAddress

4.2.8.1. Description

This procedure allows to delete a server by its address

4.2.8.2. Prototype

```
DeleteServerByAddress(IN Address VARCHAR(1000))
```

4.2.8.3. Parameters

Name	Type	Allow NULL	Description
Address	VARCHAR(1000)		Address of the server to delete

4.2.8.4. Example

```
CALL DeleteServerByAddress("10.135.20.31");
```

4.2.9. DeleteServerByServiceID

4.2.9.1. Description

This procedure allows to delete servers associated to given service

4.2.9.2. Prototype

DeleteServerByServiceID(IN Service INT)

4.2.9.3. Parameters

Name	Type	Allow NULL	Description
Service	INT		ID of the service where servers must be deleted

4.2.9.4. Example

CALL DeleteServerByServiceID(1);

5. Table DNS

5.1. Description

This table is used to identify different DNS and associate them to services

It consists of 3 attributes :

Name	Type	Allow NULL	Constraint	Description
DNS_ID	INT		Primary Key	DNS ID
DNS_Value	VARCHAR(1000)		Unique	DNS Name
DNS_Service	INT		Foreign Key	ID of the associated service

5.2. Stored procedures

5.2.1. CreateDNS

5.2.1.1. Description

This procedure allows to create a new DNS in the database

5.2.1.2. Prototype

```
CreateDNS(IN Value VARCHAR(1000), IN Service INT)
```

5.2.1.3. Parameters

Name	Type	Allow NULL	Description
Value	VARCHAR(1000)		Value of the DNS
Service	INT		ID of the associated service

5.2.1.4. Example

```
CALL CreateDNS('my.dns.com', 1);
```

5.2.2. ReadAllDNS

5.2.2.1. Description

This procedure allows to list all DNS in the database

5.2.2.2. Prototype

ReadAllDNS();

5.2.2.3. Returns

DNS list ordered alphabetically

```
MariaDB [pandore]> CALL ReadAllDNS();
```

DNS_ID	DNS_Value	DNS_Service
1	my.facebook.com	1
2	my.youtube.com	NULL

5.2.2.4. Example

CALL ReadAllDNS();

5.2.3. ReadDNSByServiceID

5.2.3.1. Description

This procedure allows to list all DNS for a given service

5.2.3.2. Prototype

`ReadDNSByServiceID(IN Service INT)`

5.2.3.3. Parameters

Name	Type	Allow NULL	Description
Service	INT		ID of the service

5.2.3.4. Returns

List of DNS associated to given service

```
MariaDB [pandore]> CALL ReadDNSByServiceID(1);
+-----+-----+-----+
| DNS_ID | DNS_Value      | DNS_Service |
+-----+-----+-----+
|      1 | my.facebook.com |           1 |
+-----+-----+-----+
```

5.2.3.5. Example

`CALL ReadDNSByServiceID(1);`

5.2.4. ReadDNSByID

5.2.4.1. Description

This procedure allows to find a DNS by its ID

5.2.4.2. Prototype

ReadDNSByID(**IN** ID **INT**)

5.2.4.3. Parameters

Name	Type	Allow NULL	Description
ID	INT		ID of the DNS

5.2.4.4. Returns

DNS found with given ID

```
MariaDB [pandore]> CALL ReadDNSByID(1);
+-----+-----+-----+
| DNS_ID | DNS_Value      | DNS_Service |
+-----+-----+-----+
|      1 | my.facebook.com |           1 |
+-----+-----+-----+
```

5.2.4.5. Example

CALL ReadDNSByID(**1**);

5.2.5. ReadServerByAddress

5.2.5.1. Description

This procedure allows to find a DNS by its value

5.2.5.2. Prototype

```
ReadDNSByValue(IN Value VARCHAR(1000))
```

5.2.5.3. Parameters

Name	Type	Allow NULL	Description
Address	VARCHAR(1000)		Value of the DNS

5.2.5.4. Returns

DNS found with given value

```
MariaDB [pandore]> CALL ReadDNSByValue('my.facebook.com');
+-----+-----+-----+
| DNS_ID | DNS_Value      | DNS_Service |
+-----+-----+-----+
|      1 | my.facebook.com |           1 |
+-----+-----+-----+
```

5.2.5.5. Example

```
CALL ReadDNSByValue('my.facebook.com');
```

5.2.6. UpdateDNS

5.2.6.1. Description

This procedure allows to update a DNS

5.2.6.2. Prototype

```
UpdateDNS(IN ID INT, IN Value VARCHAR(1000), IN Service INT)
```

5.2.6.3. Parameters

Name	Type	Allow NULL	Description
ID	INT		ID of the DNS to update
Value	VARCHAR(1000)		New DNS address
Service	INT		New service ID

5.2.6.4. Example

```
CALL UpdateDNS(1, 'my.new.dns', 2);
```

5.2.7. DeleteDNSByID

5.2.7.1. Description

This procedure allows to delete a DNS by its ID

5.2.7.2. Prototype

DeleteDNSByID(IN ID INT)

5.2.7.3. Parameters

Name	Type	Allow NULL	Description
ID	INT		ID of the DNS to delete

5.2.7.4. Example

CALL DeleteDNSByID(1);

5.2.8. DeleteDNSByValue

5.2.8.1. Description

This procedure allows to delete a DNS by its value

5.2.8.2. Prototype

```
DeleteDNSByValue(IN Value VARCHAR(1000))
```

5.2.8.3. Parameters

Name	Type	Allow NULL	Description
Value	VARCHAR(1000)		Value of the DNS to delete

5.2.8.4. Example

```
CALL DeleteDNSByValue('my.dns.todelete.com');
```


5.2.9. DeleteDNSByServiceID

5.2.9.1. Description

This procedure allows to delete DNSs associated to given service

5.2.9.2. Prototype

DeleteDNSByServiceID(IN Service INT)

5.2.9.3. Parameters

Name	Type	Allow NULL	Description
Service	INT		ID of the service where DNSs must be deleted

5.2.9.4. Example

CALL DeleteDNSByServiceID(1);

6. Table Capture

6.1. Description

This table is used to identify captures done on a device

It consists of 7 attributes :

Name	Type	Allow NULL	Constraint	Description
Capture_ID	INT		Primary Key	Capture ID
Capture_Name	VARCHAR(255)	x		Capture Name
Capture_StartTime	DATETIME			Capture start time. Format : yyyy-MM-dd HH:mm:ss
Capture_EndTime	DATETIME	x		Capture end time. Format : yyyy-MM-dd HH:mm:ss
Capture_Description	VARCHAR(1000)	x		Capture description
Capture_Interface	VARCHAR(255)	x		Capture interface (eth0, eth1, etc...)
Capture_ConnectionType	VARCHAR(255)	x		Connection type description

6.2. Stored procedures

6.2.1. CreateCapture

6.2.1.1. Description

This procedure allows to create a new capture in the database

6.2.1.2. Prototype

```
CreateCapture(IN Name VARCHAR(255), IN StartTime DATETIME, IN EndTime DATETIME,  
             IN Description VARCHAR(1000), IN Interface VARCHAR(1000), IN ConnectionType VARCHAR(1000))
```

6.2.1.3. Parameters

Name	Type	Allow NULL	Description
Name	VARCHAR(255)	x	Capture Name
StartTime	DATETIME		Capture start time. Format : yyyy-MM-dd HH:mm:ss
EndTime	DATETIME	x	Capture end time. Format : yyyy-MM-dd HH:mm:ss
Description	VARCHAR(1000)	x	Capture description
Interface	VARCHAR(255)	x	Capture interface (eth0, eth1, etc...)
ConnectionType	VARCHAR(255)	x	Connection type description

6.2.1.4. Returns

Returns the ID of the capture created

```
MariaDB [pandore]> CALL CreateCapture("My new capture", "2020-12-17 12:12:21", "2020-12-17 12:12:21", "aaa", "hhihihi", "dddddddddd");  
+-----+  
| Capture_ID |  
+-----+  
|          3 |  
+-----+
```

6.2.1.5. Example

```
CALL CreateCapture("My new capture", "2021-10-09 14:37:12", NULL,  
                  "My description", "eth0", NULL);
```

6.2.2. ReadAllCaptures

6.2.2.1. Description

This procedure allows to list all captures

6.2.2.2. Prototype

ReadAllCaptures()

6.2.2.3. Returns

List of all captures

```
MariaDB [pandore]> CALL ReadAllCaptures();
```

Capture_ID	Capture_Name	Capture_StartTime	Capture_EndTime	Capture_Description	Capture_Interface	Capture_ConnectionType
1	My new capture	2020-06-03 14:36:21	2020-06-03 17:42:45	Capture to test my application traffic	eth1	NULL
2	NULL	2021-10-10 13:40:01	NULL	NULL	NULL	NULL

6.2.2.4. Example

CALL ReadAllCaptures();

6.2.3. ReadCaptureByID

6.2.3.1. Description

This procedure allows to find a capture by its ID

6.2.3.2. Prototype

ReadCaptureByID(IN ID INT)

6.2.3.3. Parameters

Name	Type	Allow NULL	Description
ID	INT		Capture ID

6.2.3.4. Returns

The capture found with given ID

```
MariaDB [pandore]> CALL ReadCaptureByID(1);
```

Capture_ID	Capture_Name	Capture_StartTime	Capture_EndTime	Capture_Description	Capture_Interface	Capture_ConnectionType
1	My new capture	2020-06-03 14:36:21	2020-06-03 17:42:45	Capture to test my application traffic	eth1	NULL

6.2.3.5. Example

CALL ReadAllCaptures();

6.2.4. UpdateCapture

6.2.4.1. Description

This procedure allows to update a capture by its ID

6.2.4.2. Prototype

```
CREATE PROCEDURE UpdateCapture(IN ID INT, IN Name VARCHAR(255), IN StartTime DATETIME, IN EndTime DATETIME,  
IN Description VARCHAR(1000), IN Interface VARCHAR(1000), IN ConnectionType VARCHAR(1000))
```

6.2.4.3. Parameters

Name	Type	Allow NULL	Description
ID	INT		ID of the capture to update
Name	VARCHAR(255)	x	New capture name
StartTime	DATETIME		New capture start time. Format : yyyy-MM-dd HH:mm:ss
EndTime	DATETIME	x	New capture end time. Format : yyyy-MM-dd HH:mm:ss
Description	VARCHAR(1000)	x	New capture description
Interface	VARCHAR(255)	x	New capture interface (eth0, eth1, etc...)
ConnectionType	VARCHAR(255)	x	New connection type description

6.2.4.4. Example

```
CALL UpdateCapture(1, "My new capture name", "2021-10-09 14:37:12", NULL,  
"My new description", "eth0", NULL);
```

6.2.5. DeleteCaptureByID

6.2.5.1. Description

This procedure allows to delete a capture by its ID

6.2.5.2. Prototype

DeleteCaptureByID(IN ID INT)

6.2.5.3. Parameters

Name	Type	Allow NULL	Description
ID	INT		ID of the capture to delete

6.2.5.4. Example

CALL DeleteCaptureByID(1);

7. Table Capture_Request

7.1. Description

This table is used to identify requests for a specific capture

It consists of 8 attributes :

Name	Type	Allow NULL	Constraint	Description
CaptureRequest_ID	INT		Primary Key	Capture request ID
CaptureRequest_PacketSize	FLOAT			Size of the packet
CaptureRequest_Direction	TINYINT			Packet direction (0 = download, 1 = upload)
CaptureRequest_DateTime	TIMESTAMP			Request date time. Format : yyyy-MM-dd HH:mm:ss
CaptureRequest_Protocol	VARCHAR(255)			Request protocol description
CaptureRequest_Server	INT		Foreign Key	ID of the server
CaptureRequest_DNS	INT		Foreign Key	ID of the DNS
CaptureRequest_Capture	INT		Foreign Key	ID of the capture

7.2. Stored procedures

7.2.1. CreateRequest

7.2.1.1. Description

This procedure allows to create a new capture request in the database

7.2.1.2. Prototype

```
CreateRequest(IN PacketSize FLOAT, IN Direction TINYINT(1), IN Protocol VARCHAR(255),  
IN Server INT, IN DNS INT, IN Capture INT)
```

7.2.1.3. Parameters

Name	Type	Allow NULL	Description
PacketSize	FLOAT		Size of the packet
Direction	TINYINT		Packet direction (0 = download, 1 = upload)
Protocol	VARCHAR(255)		Request protocol description
Server	INT	x	ID of the server
DNS	INT	x	ID of the DNS
Capture	INT		ID of the capture

7.2.1.4. Example

```
CALL CreateRequest(17, 0, 'TCP', 1, 1, 1);
```

7.2.2. CreateRequestString

7.2.2.1. Description

This procedure allows to create a new capture request in the database without knowing the id of the DNS and the server but only their values.

If DNS is not null and doesn't exist in the database, new DNS is added in DNS table.

If Server is not null and doesn't exist in the database, new server is added in Server table.

If DNS is null or empty, request DNS will also be null and nothing will be added in DNS table.

If Server is null or empty, request Server will also be null and nothing will be added in Server table.

7.2.2.2. Prototype

```
CreateRequestString(IN PacketSize FLOAT, IN Direction TINYINT(1), IN Protocol VARCHAR(255),  
IN Server VARCHAR(1000), IN DNS VARCHAR(1000), IN Capture INT)
```

7.2.2.3. Parameters

Name	Type	Allow NULL	Description
PacketSize	FLOAT		Size of the packet
Direction	TINYINT		Packet direction (0 = download, 1 = upload)
Protocol	VARCHAR(255)		Request protocol description
Server	VARCHAR(1000)	x	Server address
DNS	VARCHAR(1000)	x	DNS value
Capture	INT		ID of the capture

7.2.2.4. Example

```
CALL CreateRequestString(17, 0, 'TCP', '10.135.20.32', 'dns.name.com', 1);
```

7.2.3. ReadAllRequests

7.2.3.1. Description

This procedure allows to list all capture requests

7.2.3.2. Prototype

`ReadAllRequests(IN Details TINYINT(1))`

7.2.3.3. Parameters

Name	Type	Allow NULL	Description
Details	TINYINT(1)		0 = returns list of requests 1 = returns list of requests with join with DNS and server tables

7.2.3.4. Returns

List of all capture requests (details = 0)

```
MariaDB [pandora]> CALL ReadAllRequests(0);
```

CaptureRequest_ID	CaptureRequest_PacketSize	CaptureRequest_Direction	CaptureRequest_DateTime	CaptureRequest_Protocol	CaptureRequest_Server	CaptureRequest_DNS	CaptureRequest_Capture
2	985	1	2021-10-28 11:38:04	IPv4	2	2	1
1	17	1	2021-10-28 11:37:42	IPv4	1	1	1

List of all capture requests with DNS and server details (details = 1)

```
MariaDB [pandora]> CALL ReadAllRequests(1);
```

CaptureRequest_ID	CaptureRequest_PacketSize	CaptureRequest_Direction	CaptureRequest_DateTime	CaptureRequest_Protocol	CaptureRequest_Server	CaptureRequest_DNS	CaptureRequest_Capture	Server_ID	Server_Address	Server_Service	DNS_ID	DNS_Value	DNS_Service
2	985	1	2021-10-28 11:38:04	IPv4	2	2	1	2	123.12.53.32	NULL	2	10.157.3.65	NULL
1	17	1	2021-10-28 11:37:42	IPv4	1	1	1	1	10.32.53.32	NULL	1	10.12.3.45	NULL

7.2.3.5. Example

`CALL ReadAllRequests(1);`

7.2.4. ReadRequestsByCaptureID

7.2.4.1. Description

This procedure allows to list all requests for a given capture

7.2.4.2. Prototype

`ReadRequestsByCaptureID(IN Capture INT, IN Details TINYINT(1))`

7.2.4.3. Parameters

Name	Type	Allow NULL	Description
Capture	INT		ID of the capture
Details	TINYINT(1)		0 = returns list of requests 1 = returns list of requests with join with DNS and server tables

7.2.4.4. Returns

List of all capture requests for the given capture ID (details = 0)

```
MariaDB [pandora]> CALL ReadRequestsByCaptureID(1, 0);
```

CaptureRequest_ID	CaptureRequest_PacketSize	CaptureRequest_Direction	CaptureRequest_DateTime	CaptureRequest_Protocol	CaptureRequest_Server	CaptureRequest_DNS	CaptureRequest_Capture
1	17	1	2021-10-28 11:37:42	IPv4	1	1	1
2	985	1	2021-10-28 11:38:04	IPv4	2	2	1

List of all capture requests for the given capture ID with DNS and server details (details = 1)

```
MariaDB [pandora]> CALL ReadRequestsByCaptureID(1, 1);
```

CaptureRequest_ID	CaptureRequest_PacketSize	CaptureRequest_Direction	CaptureRequest_DateTime	CaptureRequest_Protocol	CaptureRequest_Server	CaptureRequest_DNS	CaptureRequest_Capture	Server_ID	Server_Address	Server_Service	DNS_ID	DNS_Value	DNS_Service
1	17	1	2021-10-28 11:37:42	IPv4	1	1	1	1	10.32.53.32	NULL	1	10.12.3.45	NULL
2	985	1	2021-10-28 11:38:04	IPv4	2	2	1	2	123.12.53.32	NULL	2	10.157.3.65	NULL

7.2.4.5. Example

`CALL ReadRequestsByCaptureID(2, 1);`

7.2.5. ReadRequestByID

7.2.5.1. Description

This procedure allows to find a request by its ID

7.2.5.2. Prototype

ReadRequestByID(**IN** ID **INT**, **IN** Details **TINYINT**(1))

7.2.5.3. Parameters

Name	Type	Allow NULL	Description
ID	INT		ID of the request to update
Details	TINYINT(1)		0 = returns list of requests 1 = returns list of requests with join with DNS and server tables

7.2.5.4. Returns

The request found for the given request ID (details = 0)

```
MariaDB [pandora]> CALL ReadRequestByID(1, 0);
```

CaptureRequest_ID	CaptureRequest_PacketSize	CaptureRequest_Direction	CaptureRequest_DateTime	CaptureRequest_Protocol	CaptureRequest_Server	CaptureRequest_DNS	CaptureRequest_Capture
1	17	1	2021-10-28 11:37:42	IPv4	1	1	1

The request found for the given request ID with DNS and server details (details = 1)

```
MariaDB [pandora]> CALL ReadRequestByID(1, 1);
```

CaptureRequest_ID	CaptureRequest_PacketSize	CaptureRequest_Direction	CaptureRequest_DateTime	CaptureRequest_Protocol	CaptureRequest_Server	CaptureRequest_DNS	CaptureRequest_Capture	Server_ID	Server_Address	Server_Service	DNS_ID	DNS_Value	DNS_Service
1	17	1	2021-10-28 11:37:42	IPv4	1	1	1	1	10.32.53.32	NULL	1	10.12.3.45	NULL

7.2.5.5. Example

CALL ReadAllRequestByID(1, 1);

7.2.6. UpdateRequest

7.2.6.1. Description

This procedure allows to update an existing capture request

7.2.6.2. Prototype

```
UpdateRequest(IN ID INT, IN PacketSize FLOAT, IN Direction TINYINT(1), IN DateTime TIMESTAMP,  
IN Protocol VARCHAR(255), IN Server INT, IN DNS INT, IN Capture INT)
```

7.2.6.3. Parameters

Name	Type	Allow NULL	Description
ID	INT		ID of the request to update
PacketSize	FLOAT		New packet size
Direction	TINYINT		New packet direction (0 = download, 1 = upload)
DateTime	TIMESTAMP		New request date time. Format : yyyy-MM-dd HH:mm:ss
Protocol	VARCHAR(255)		New protocol description
Server	INT		ID of the new server
DNS	INT		ID of the new DNS
Capture	INT		ID of the new capture

7.2.6.4. Example

```
CALL CreateRequest(1, 17, 0, '2021-11-27 14:08:30', 'TCP', 1, 1, 1);
```

7.2.7. DeleteRequestByID

7.2.7.1. Description

This procedure allows to delete a capture request by its ID

7.2.7.2. Prototype

DeleteRequestByID(**IN** ID **INT**)

7.2.7.3. Parameters

Name	Type	Allow NULL	Description
ID	INT		ID of the capture request to delete

7.2.7.4. Example

CALL DeleteRequestByID(**1**);

7.2.8. DeleteRequestByCaptureID

7.2.8.1. Description

This procedure allows to delete all requests for a given capture

7.2.8.2. Prototype

DeleteRequestByCaptureID(IN Capture INT)

7.2.8.3. Parameters

Name	Type	Allow NULL	Description
Capture	INT		ID of the capture

7.2.8.4. Example

CALL DeleteRequestByCaptureID(1);