Virtual DDBMS Developer Documentation
Draft v0.8.0.0

1. System Requirements

The project is mainly developed using Fedora 8 (Werewolf) with these packages installed:

- GNU binutils[1] version 2.17.50.0.18
- GNU Bash[1] version 3.2.25
- GNU Make[1] version 3.8.1
- GNU GCC[2] version 4.1.2
- GNU libc[1] version 2.7 + development package
- Libgcrypt[1] version 1.2.4 + development package
- MySQL[2] version 5.0.45 (14.12) + development package
- MySQL++[2] version 3.0.0 + development package
- PostgreSQL[2,4] version 8.2.5 + development package
- FreeTDS[2,4] version 0.82 + development package
- Oracle XE Client[2,4] version 10.2.0.1
- OCILib[2,4] version 3.0.1 + development package
- UnixODBC[2,4] version 2.2.12 + development package
- PHP[2,4] version 5.2.4 + development package
- Sun JDK[3,4] version 1.6.0_01

The project is also built-able using Rocks 4.2.1 (Cydonia) with these packages installed:

- GNU binutils[1] version 2.15.92.0.2
- GNU Bash[1] version 3.0.15
- GNU Make[1] version 3.8.0
- GNU GCC[2] version 4.3.2 (manually upgraded from the default version 3.4.6)
- GNU libc[1] version 2.3.4 + development package
- Libgcrypt[1] version 1.2.0 + development package
- MySQL[2] version 4.1.20 (14.7) + development package
- MySQL++[2] version 3.0.0 + development package
- PostgreSQL[2,4] version 7.4.13 + development package
- FreeTDS[2,4] version 0.82 + development package
- UnixODBC[2,4] version 2.2.11 + development package
- PHP[2,4] version 5.1.6 + development package
- Sun JDK[3,4] version 1.6.0_01

Note:

1. Newer and slightly older versions should still work.
2. Newer and slightly older versions should still work as long as the major version numbers are still the same.
3. Only needed to build the Java network client library, generally newer versions are preferred.
4. Optional.

GCC older than version 4.0.0 may not be able to build the library.

In order to build the .NET network client library, Microsoft® Visual Studio® with .NET Framework version 2.x is needed.

In order to deploy and test the libraries and applications correctly, minimum three computers with a properly configured MySQL server are recommended. Although one computer will still usable, but it would be impossible to test the distributed nature of the database in this way.

Installing the dependencies will be OS and/or distribution specific. Hence, this documentation will not discuss on how to install those dependencies. Please consult your OS and/or distribution documentations.

2. Building and Installing

The project tree are divided according to the sub-system:

- Linux C++ core engine : **`ddbms/engine`**
- Linux C++ network server daemon : **`ddbms/dbserver`**
- Linux C++ network client library : **`ddbms/dbclient`**
- Java network client library : **`ddbms/dbclient-java`**
- .NET network client library : **`ddbms/dbclient-dotnet`**

Edit the file **`Makefile.config`** to adjust the build settings according your needs.

For the Linux C++ library/application, executing these commands from the main VDDBMS source directory:

```
cd engine
make install
cd ../dbserver
make install
cd ../dbclient
make install
```

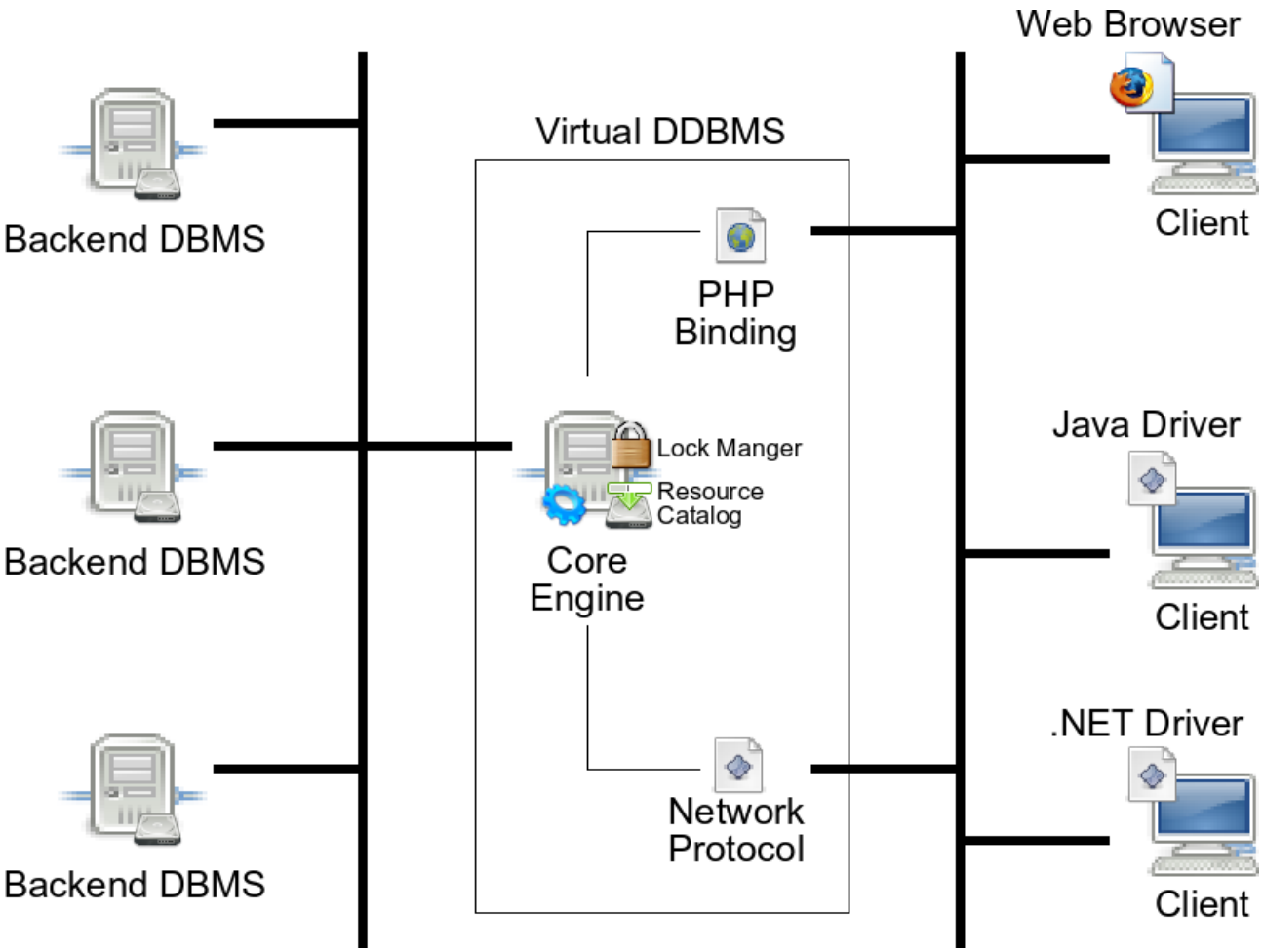
will build and install the header, library, and binary files to **`/usr/local/include/vddbms`**, **`/usr/local/include/vddbmsnc`**, **`/usr/local/lib`**, and **`/usr/local/bin`**. If you got errors, make sure that the dependencies mentioned before are already fulfilled.

Note that you will need to execute the **`make install`** as **`root`** or give the current user **`sudo`** capability.

For the Java and .NET client libraries, pre-compiled **`.jar`** and **`.dll`** files are provided.

3. Configuring the System

Currently the architecture of the system is as follow:



As configured in the file **`ddbms/engine/DDBMSSuperDef.h`**, the engine will connect to the backend DBMS using user **`gcl_ddbms_user`** and password **`gcl_ddbms_user`** (you can change them). The clients can connect to the Virtual DDBMS (VDDBMS) by using either:

- HTTP port 80
- Our custom client network driver at port 25001 (as configured in the file **`ddbms/dbserver/gclddbmsd.cc`**).

Hence, make sure that no firewall blocking those ports.

Configuring the Slave Nodes

For all the slave nodes (the data sources), the VDDBMS will expect the database named **`gcl_ddbms_data`** (you may use different name if you like) to be exist and accessible by the user that will be used by the VDDBMS.

For MySQL, it can be done by using MySQL console and entering this command (in one line):

```
GRANT ALL PRIVILEGES ON gcl_ddbms_data_1.* TO 'my_user'@'location'
IDENTIFIED BY 'my_password' WITH GRANT OPTION;

CREATE DATABASE gcl_ddbms_data_1;
```

Note:

- Replace ***gcl_ddbms_data_1*** with different name if you like.
- Replace ***location*** with the IP or fully qualified hostname of the master node.

For PostgreSQL, execute these commands in console (each in one line):

```
/usr/bin/createuser --no-superuser --no-createrole –createdb
--login --pwprompt –encrypted my_user
```

<when prompted, set the password of ***my_user***>

```
/usr/bin/createdb gcl_ddbms_data_1
```

then edit **/var/lib/pgsql/data/pg_hba.conf** and add this line:

```
host gcl_ddbms_data_1 my_user xxx.xxx.xxx.xxx/32 md5
```

and finally restart the PostgreSQL service.

Note:

- Replace ***gcl_ddbms_data_1*** with different name if you like.
- Replace ***xxx.xxx.xxx.xxx*** with the IP of the master node.

In case you plan to store large data, please make sure that the backend DBMS supports it. For example, in MySQL, it can be done by setting the **max_allowed_packet** variable inside the MySQL daemon configuration file (usually **/etc/my.cnf**) in the **[mysqld]** section.

Configuring the Master Node

The master node (the VDDBMS) needs a properly configurable database for its resource catalog (named **gcl_ddbms_system**). The database currently must be in the same host (localhost). The DBMS needs to allow the VDDBMS user (**gcl_ddbms_user**) to access the needed database.

For MySQL, it can be done by using MySQL console and entering this command (in one line):

```
GRANT ALL PRIVILEGES ON gcl_ddbms_system.* TO 'my_user'@'localhost'
IDENTIFIED BY 'my_password' WITH GRANT OPTION;

GRANT ALL PRIVILEGES ON gcl_ddbms_system.* TO 'my_user'@'location'
IDENTIFIED BY 'my_password' WITH GRANT OPTION;

CREATE DATABASE gcl_ddbms_system;
```

Note:

- Replace ***location*** with the IP or fully qualified hostname of the master node).

For PostgreSQL, one can execute in console (each in one line):

```
/usr/bin/createuser --no-superuser --no-createrole –createdb
--login --pwprompt –encrypted my_user
```

<when prompted, set the password of ***my_user***>

```
/usr/bin/createdb gcl_ddbms_system
```

then edit **`/var/lib/pgsql/data/pg_hba.conf`** and add this line:

```
local gcl_ddbms_system my_user md5
```

and finally restart the PostgreSQL service.

You can change the strings database name, user name, and passwords to anything you like, but do not forget to also change the constants defined in **`GCLDDBMS.h`** :

- GCL_DDBMS_SYSTEM_USER_NAME
- GCL_DDBMS_SYSTEM_USER_PASSWORD
- GCL_DDBMS_SYSTEM_DATABASE_NAME

It is recommended that the resource catalog (the **`gcl_ddbms_system`** database) is deployed using MySQL.

In the future, when Replicated Resource Catalog and Distributed Lock Manager have been implemented, there will be more than master nodes that can be installed.

Configuring the VDDBMS Server in the Master Node

The configuration file will be installed in **`/etc/gclddbmsd.conf`**. However when the VDDBMS server is run in non-daemon mode for debugging purpose, it will read the configuration file in **`ddbms/dbserver/gclddbmsd.conf_`**. The table below explains the entries in the configuration file:

Parameter	Format	Default Value	Purpose
`lms_port`	`integer`	`25000`	The port number that the lock manager server is listening to.
`service_port`	`integer`	`25001`	The port number that the server will listen to.
`max_connection`	`integer`	`1000`	The number of simultaneous connection allowed (zero or empty means use default).
`login_timeout`	`integer`	`90`	How long (in seconds) the server will wait for valid 'Request for Login' before disconnecting (zero or empty means use default).
`idle_timeout`	`integer`	`10`	How long (in minutes) the server will wait for an idle client before disconnecting (zero or empty means use default).
`host_allow`	`list of text`	`N/A`	List of host IP or domain name allowed to connect with the server (empty means none allowed).
`host_deny`	`list of text`	`N/A`	List of host IP or domain name denied from connecting with the server (empty means none denied). The list of **`host_deny`** has higher priority than the list of **`host_allow`**.
`primary_rescat_type`	`text`	`mysql`	Type of the primary resource catalog database in localhost (empty means use default).
`primary_rescat_port`	`integer`	`3306`	Port of the primary resource catalog database in localhost (zero or empty means use default).
`add_user`	`name:password:mapped_id:acl`	`N/A`	Defining a user, example: **`owner:magic:xxxx:11111111111`** means: user name = owner password = magic mapped ID = xxxx ACL = all operations to all databases owned by xxxx Another example: **guest`::xxxx:00000000000`** means: user name = guest password = <no password> mapped ID = xxxx ACL = reead only to all databases owned by xxxx

The mapped ID is four base-64 characters determining the database owner. The choice of characters are:

`0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ+-`

The ACL is a 11-bit wide bitset, read from left to right, controlling these features:

1. canUpdateRecordsInTables
2. canInsertDeleteRecordsFromTables
3. canShowTables
4. canCreateDeleteTables
5. canShowDatabases
6. canCreateDeleteDatabases
7. canGrantAccess
8. canShowResources
9. canAddRemoveResources

Note that currently the engine does not handle user management. Therefore, it is up to you to define the correct **`mapped_id`** (ID of database owner) for each of the user.
Code cleanups
Below is a full example of the configuration file:

```
#
# An example of server configuration file (comment is started by #)
#
lms_port            = 25000
service_port        = 25001
max_connection      = 250
idle_timeout        = 10
host_allow          = localhost, trusted.com
host_deny           = 10.0.0.1, cracker.com
primary_rescat_type = mysql
primary_rescat_port = 3306
add_user            = owner:magic:xxxx:11111111111
add_user            = guest::xxxx:00000000000
```

In the future, more configuration entries may be added.

Please make sure that the local DBMS used as the resource catalog is able to accept connections for at least as many as the number specified in **`max_connection`**. If direct-engine-access local API will be also used in the VDDBMS server, the resource catalog DBMS will need to be configured to accept much more connections. For example, in MySQL, it can be done by setting the **`max_connections`** variable inside the MySQL daemon configuration file (such as **`/etc/my.cnf`**).

Starting the Daemons

To start the daemons, execute the commands:

```
lmserverd start
gclddbmsd start
```

To find what parameters (commands) supported by the daemons, just execute the command without the parameter. They will display the supported commands.

4. Supported SQL

Standard SQL Commands/Statements Currently Supported by the Engine

```
CREATE DATABASE [IF NOT EXISTS] db_name

DROP DATABASE [IF EXISTS] db_name

USE [DATABASE] db_name


CREATE TABLE [IF NOT EXISTS] [db_name.]table_name
(
       col_name <col_type> [NOT NULL | NULL] [UNIQUE] [PRIMARY KEY | KEY]
    [, col_name <col_type> [NOT NULL | NULL] [UNIQUE] [PRIMARY KEY | KEY] ]...

    [, [FULLTEXT] INDEX index_name (col_name[,...]) ]...
)

DROP TABLE [IF EXISTS] [db_name.]table_name [, [db_name.]table_name ]...

RENAME TABLE    [db_name.]table_name TO [db_name.]new_table_name
             [, [db_name.]table_name TO [db_name.]new_table_name ]...

DESCRIBE [db_name.]table_name

SHOW | COUNT TABLES


INSERT INTO [db_name.]table_name [(col_name[,...])] VALUES (<expr>[,...])

DELETE
FROM                   [db_name.]table_name
                    [, [db_name.]table_name ]...
[USING                 [db_name.]table_name
                    [, [db_name.]table_name ]... ]
[WHERE <where_def>]

UPDATE [db_name.]table_name [, [db_name.]table_name ]...
    SET                    col_name = <expr>
                        [, col_name = <expr> ]...
    [WHERE <where_def>]

SELECT { * | COUNT(*) | col_name[,...] }
    FROM                   [db_name.]table_name
                        [, [db_name.]table_name ]...
    [WHERE <where_def>]
    [ORDER BY              col_name [ASC | DESC]
                        [, col_name [ASC | DESC] ]... ]
    [LIMIT row_count]
```

```
<col_type>
    <data_type> [(<param>[,...])] [<modifier>]...
examples:
    INTEGER        UNSIGNED
    FLOAT   (4, 4) ZEROFILL
    CHAR    (20)   BINARY
    TEXT           ASCII
    DATE           UNICODE

<where_def>
    [[db_name.]table_name.]col_name { IS [NOT] {NULL | FALSE | TRUE} |
                                      [NOT] BETWEEN min AND max      |
                                      [NOT] IN (value[,...])         |
                                      <operator> <expr>
                                    }
    [ {AND | OR } [[db_name.]table_name.]col_name { IS [NOT] {NULL | FALSE | TRUE} |
                                                    [NOT] BETWEEN min AND max      |
                                                    [NOT] IN (value[,...])         |
                                                    <operator> <expr>
                                                  }
    ]...

<operator>
    =   <=>    <>   !=   <=   <   >=   >
```

Extended/Custom SQL Commands/Statements Currently Supported by the Engine

Simple SQL

```
SHOW | COUNT RESOURCES

SHOW | COUNT DATABASES

SHOW | COUNT PROCEDURES


CREATE PROCEDURE <sp_name> ([<param_def1>,...]) <body_def>

DROP PROCEDURE [IF EXISTS] <sp_name>

DISPLAY PROCEDURE <sp_name>

CALL <sp_name> ([<param1>,...])


SET @system_var_name = constant


GRANT READ PRIVILEGE ON [db_name1.]table_name TO 'user_id' AS db_name2.new_table_name
    example:
        user 'xxxx' who owns 'Sales.Soap' can type:
            GRANT READ PRIVILEGE ON Sales.Soap TO 'yyyy' AS Watcher.WSoap
        to grant user 'yyyy' read access to 'Sales.Soap' using alias table 'Watcher.WSoap'.


ADD RESOURCE 'type', 'host', port, 'db_name', 'user', 'password', 'odbc_opts', 'description'

REMOVE RESOURCE 'resource_id'

REPLACE RESOURCE 'resource_id'
        WITH 'type', 'host', port, 'db_name', 'user', 'password', 'odbc_opts', 'description'
```

Support for Replication and Fragmentation

```
CREATE TABLE ... () { REPLICATE <rep_def>   |
                      HFRAGMENT <hfrag_def> |
                      VFRAGMENT <vfrag_def> |
                      XFRAGMENT <xfrag_def>   }

    REPLICATE (res_id[:res_id]...)

    XFRAGMENT (<xml_schema_file>)

    VFRAGMENT    (col_name[:col_name]...)(res_id[:res_id]...)
              [, (col_name[:col_name]...)(res_id[:res_id]...) ]...

              for vertical fragmentation, it is an error if there is any column which is not
              covered by any of the rules.

    HFRAGMENT    (<expr>)(res_id[:res_id]...)
              [, (<expr>)(res_id[:res_id]...) ]
              [ OTHERS { USE <ref_to_prev_rule> | IS ERROR | (res_id[:res_id]...) } ]

        <ref_to_prev_rule> is index (start from 1) to the previously defined rules

        <expr>
            { <value> <operator> col_name <operator> <value> |
                                 col_name <operator> <value> |
                                 col_name IS { ODD | EVEN }
            }
        examples:
            1 <= Year <= 2
            Year >= 3
            Year IS ODD

    Note: * Currently, XFRAGMENT and VFRAGMENT are not yet implemented.
          * For SQL 'UPDATE' command, 'updating the field that defines the horizontal
            fragmentation key' is not yet supported.
```

Support for Blob Table

```
CREATE BLOB TABLE [IF NOT EXISTS] [db_name.]table_name WITH { NUMERIC | STRING } ID
    { REPLICATE <rep_def> | HFRAGMENT <hfrag_def> }

    REPLICATE (res_id[:res_id]...)

    HFRAGMENT    (<id_expr>)(res_id[:res_id]...)
              [, (<id_expr>)(res_id[:res_id]...) ]
              [ OTHERS { USE <ref_to_prev_rule> | IS ERROR | (res_id[:res_id]...) } ]

        <ref_to_prev_rule> is index (start from 1) to the previously defined rules

        <id_expr>
            { <value> <operator> ID <operator> <value> |
                                 ID <operator> <value> |
                                 ID IS { ODD | EVEN }
            }
        examples:
            1 <= ID <= 2
            ID >= 3
            ID IS ODD

    Note: Currently, 'NUMERIC' ID will be implemented using 'INTEGER' while
          'STRING' ID using 'CHAR(32)'.


LOAD BLOB BUFFER WITH { NULL | DATA blob_data }

    Note: * After 'DATA', there must be exactly one white-space before 'blob_data'.
            Anything after the white-space will be considered as the data of the blob).
          * In case of multi-statement, this statement must be put as the last one.


INSERT BLOB INTO [db_name.]table_name SET ID = new_id, DATA = { BUFFER | NULL }


DELETE BLOB FROM [db_name.]table_name [ WHERE ID { IN (value[,...]) |
                                                   <operator> value
                                                 }
                                      ]
    <operator>
        =   <=>    <>   !=   <=   <   >=   >


UPDATE BLOB IN [db_name.]table_name
    SET     [ ID   = new_id            ]
        [,] [ DATA = { BUFFER | NULL } ]
  [ WHERE ID { IN (<value>[,...]) |
               <operator> <value>
             }
  ]

    Note: * The clause 'SET ID TO new_id' is only valid with the clause 'WHERE ID = value'.
          * Currently, 'updating the field that defines the horizontal fragmentation key
            (the blob ID)' is not supported yet.


SELECT BLOB FROM [db_name.]table_name WHERE ID = <value>

    Note: This statement will always return a rowset with two rows.
```

Support for Aggregate Table

```
<col_def>

    col_name <col_type> [NOT NULL | NULL] [UNIQUE] [PRIMARY KEY | KEY]


CREATE AGGREGATE TABLE [IF NOT EXISTS] [db_name.]table_name
FROM res1_id.ptable1_name[:res2_id.ptable2_name]...
COLUMN MAP
(
    <col_def> => (pcol1_name[:pcol2_name]...)
    [,...]
)
```

```
    Examples:
        CREATE AGGREGATE TABLE MyDB.MyStuff
            FROM aaaa.Goods
            COLUMN MAP
            (
                ID   INT  NOT NULL PRIMARY KEY => (Code),
                Name TEXT NOT NULL             => (Name),
                Note TEXT     NULL             => (Note)
            )

        CREATE AGGREGATE TABLE MyDB.MyStuff
            FROM aaaa.Goods:bbbb.Things
            COLUMN MAP
            (
                ID   INT  NOT NULL PRIMARY KEY => (Code:Type),
                Name TEXT NOT NULL             => (Name:Desc),
                Note TEXT     NULL             => (Note:Memo)
            )


CREATE AGGREGATE TABLE [IF NOT EXISTS] [db_name.]table_name
HFRAGMENT ON    <expr1>
             [, <expr2>]...
FROM            res1a_id.ptable1a_name[:res1b_id.ptable1b_name]
             [, res2a_id.ptable2a_name[:res2b_id.ptable2b_name] ]...
COLUMN MAP
(
    <col_def> =>   (pcol1a_name[:pcol1b_name]...)
                 [ (pcol2a_name[:pcol2b_name]...) ]...
    [,...]
)

    Examples:
        CREATE AGGREGATE TABLE MyDB.MyStuff
            HFRAGMENT ON ID IS ODD  , ID IS EVEN
            FROM         aaaa.Goods , cccc.Goods
            COLUMN MAP
            (
                ID   INT  NOT NULL PRIMARY KEY => (Code)(Code),
                Name TEXT NOT NULL             => (Name)(Name),
                Note TEXT     NULL             => (Note)(Note)
            )

        CREATE AGGREGATE TABLE MyDB.MyStuff
            HFRAGMENT ON ID < 0     , ID > 0     , OTHERS IS ERROR
            FROM         aaaa.Goods , cccc.Goods
            COLUMN MAP
            (
                ID   INT  NOT NULL PRIMARY KEY => (Code)(Code),
                Name TEXT NOT NULL             => (Name)(Name),
                Note TEXT     NULL             => (Note)(Note)
            )

        CREATE AGGREGATE TABLE MyDB.MyStuff
            HFRAGMENT ON ID < 0     , ID > 0     , OTHERS USE 2
            FROM         aaaa.Goods , cccc.Goods
            COLUMN MAP
            (
                ID   INT  NOT NULL PRIMARY KEY => (Code)(Code),
                Name TEXT NOT NULL             => (Name)(Name),
                Note TEXT     NULL             => (Note)(Note)
            )

        CREATE AGGREGATE TABLE MyDB.MyStuff
            HFRAGMENT ON ID < 0     , ID > 0     , OTHERS
            FROM         aaaa.Goods , cccc.Goods , eeee.Things
            COLUMN MAP
            (
                ID   INT  NOT NULL PRIMARY KEY => (Code)(Code)(Type),
                Name TEXT NOT NULL             => (Name)(Name)(Desc),
                Note TEXT     NULL             => (Note)(Note)(Memo)
            )
```

```
        CREATE AGGREGATE TABLE MyDB.MyStuff
            HFRAGMENT ON ID IS ODD              , ID IS EVEN
            FROM         aaaa.Goods:bbbb.Things , cccc.Goods:dddd.Things
            COLUMN MAP
            (
                ID   INT  NOT NULL PRIMARY KEY => (Code:Type)(Code:Type),
                Name TEXT NOT NULL             => (Name:Desc)(Name:Desc),
                Note TEXT     NULL             => (Note:Memo)(Note:Memo)
            )


CREATE AGGREGATE TABLE [IF NOT EXISTS] [db_name.]table_name
VFRAGMENT ON    col1_name[:...]
             [, col2_name[:...] ]...
FROM            res1a_id.ptable1a_name[:res1b_id.ptable1b_name]
             [, res2a_id.ptable2a_name[:res2b_id.ptable2b_name] ]...
COLUMN MAP
(
       <col1_def> => (pcol1a_name[:pcol1b_name]...)
    [, <col2_def> => (pcol2a_name[:pcol2b_name]...) ]...
)
    Examples:
        CREATE AGGREGATE TABLE MyDB.MyStuff
            VFRAGMENT ON ID:Name     , ID:Note:Optz
            FROM         aaaa.Goods1 , cccc.Goods2
            COLUMN MAP
            (
                ID   INT  NOT NULL PRIMARY KEY => (Code)(Code),
                Name TEXT NOT NULL             => (Name)(Name),
                Note TEXT     NULL             => (Note)(Note),
                Optz TEXT     NULL             => (Optz)(Optz)
            )

        CREATE AGGREGATE TABLE MyDB.MyStuff
            VFRAGMENT ON ID:Name                   , ID:Note:Optz
            FROM         aaaa.Goods1:bbbb.Things1 , cccc.Goods2:dddd.Things2
            COLUMN MAP
            (
                ID   INT  NOT NULL PRIMARY KEY => (Code:Type)(Code:Type),
                Name TEXT NOT NULL             => (Name:Desc)(Name:Desc),
                Note TEXT     NULL             => (Note:Memo)(Note:Memo),
                Optz TEXT     NULL             => (Optz:Optz)Optz:Optz)
            )


CREATE AGGREGATE TABLE [IF NOT EXISTS] [db_name.]table_name
    XFRAGMENT USING (<xml_schema_file>)


Note: * Currently, XFRAGMENT and VFRAGMENT are not yet implemented.
```

Support for Aggregate Blob Table

```
CREATE AGGREGATE BLOB TABLE [IF NOT EXISTS] [db_name.]table_name WITH { NUMERIC | STRING } ID
[ HFRAGMENT ON    <id_expr1>
               [, <id_expr2>]... ]
  FROM            res1a_id.ptable1a_name[:res1b_id.ptable1b_name]
                  [, res2a_id.ptable2a_name[:res2b_id.ptable2b_name] ]...
  MAP             ID   => (pcol1a_name[:pcol1b_name]...) [(pcol2a_name[:pcol2b_name]...)]...
                  DATA => (pcol1a_name[:pcol1b_name]...) [(pcol2a_name[:pcol2b_name]...)]...

    Examples:
        CREATE AGGREGATE BLOB TABLE MyDB.MyBlob WITH NUMERIC ID
            FROM aaaa.Whatever
            MAP  ID   => (Code   ),
                 DATA => (Content)

        CREATE AGGREGATE BLOB TABLE MyDB.MyBlob WITH NUMERIC ID
            FROM aaaa.Whatever:bbbb.Whatever
            MAP  ID   => (Code   :Code   ),
                 DATA => (Content:Content)

        CREATE AGGREGATE BLOB TABLE MyDB.MyBlob WITH NUMERIC ID
            HFRAGMENT ON ID IS ODD     , ID IS EVEN
            FROM         aaaa.Whatever , cccc.Whatever
            MAP          ID   => (Code   )(Code   ),
                         DATA => (Content)(Content)
```

```
CREATE AGGREGATE BLOB TABLE MyDB.MyBlob WITH NUMERIC ID
    HFRAGMENT ON ID IS ODD                  , ID IS EVEN
    FROM         aaaa.Whatever:bbbb.Whatever , cccc.Whatever:dddd.Whatever
    MAP          ID   => (Code   :Code   )(Code   :Code   ),
                 DATA => (Content:Content)(Content:Content)
```

Supported Date/Time Formats

Below are the date and time formats supported by the engine:

```
Date          : YYYY/MM/DD
Time          : hh:mm:ss ±hhmm
Date and time : YYYY/MM/DD hh:mm:ss ±hhmm
```

Note that the GMT offset (`±hhmm`) is mandatory.

5. System Variables

Name	Unit	Functions	Default Value	Minimum Value	Maximum Value
`@sp_pool_interval`	seconds	Time interval for the checking of the completion of stored procedure execution.	10	1	30
`@sp_timeout_limit`	seconds	Timeout for the completion of stored procedure execution. In the reality, the total timeout may be as long as the total value of `@sp_timeout_limit` and `@sp_pool_interval`.	90	10	600

6. Building Client Application using Direct-Engine-Call API

This kind of application must be deployed in the same computer with the VDDMS master node.

Below is an example C++ application:

```
#include <iostream>
using namespace std;

#include <DDBMSConnection.h>
using namespace gclddbms;

int main()
{
    DBRow           dbr;
    DDBMSConnection ddbms("xxxx", "mysql", 3306,
                          true, true, true, true, true, true, true, true, true);

    const string& res1ID = ddbms.addResource("mysql",
                                             "slave1.net",
                                             3306,
                                             "db_name",
                                             "my_user",
                                             "my_user",
                                             "",
                                             "Grid Portal");
    const string& res2ID = ddbms.addResource("mysql",
                                             "slave2.net",
                                             3306,
                                             "db_name",
                                             "my_user",
                                             "my_user",
                                             "",
                                             "MyProxy Server");

    ddbms.execSQL("SHOW RESOURCES");
    while(ddbms.fetchNextRow(dbr)) {
        for(size_t i = 0; i < dbr.size(); ++i) cerr << dbr[i] << "\t";
        cerr << endl;
    }
    cerr << endl;

    ddbms.execSQL("CREATE DATABASE TestDB");
    ddbms.execSQL("USE TestDB");

    ddbms.execSQL("CREATE TABLE TestTB (ID INT, Name TEXT, Address TEXT) "
                  "HFRAGMENT (ID IS ODD )(" + res1ID + "),               "
                  "          (ID IS EVEN)(" + res2ID + ")               ");

    ddbms.execSQL("INSERT INTO TestTB VALUES (0, 'Name 0', 'Address 0')");
    ddbms.execSQL("INSERT INTO TestTB VALUES (1, 'Name 1', 'Address 1')");
    ddbms.execSQL("INSERT INTO TestTB VALUES (2, 'Name 2', 'Address 2')");
    ddbms.execSQL("INSERT INTO TestTB VALUES (3, 'Name 3', 'Address 3')");

    ddbms.execSQL("SELECT * FROM TestTB");
    while(ddbms.fetchNextRow(dbr)) {
        for(size_t i = 0; i < dbr.size(); ++i) cerr << dbr[i] << "\t";
        cerr << endl;
    }
    cerr << endl;

    ddbms.execSQL("DROP DATABASE TestDB");

    ddbms.removeResource(res1ID);
    ddbms.removeResource(res2ID);
}
```

Please refer to the C++ specific documentation for the API.

Below is an example PHP application:

```
<?php
    try {
        session_start();
        if(!isset($_SESSION["ddbms_session_id"])) {
            $_SESSION["ddbms_session_id"] = session_id();
        }
        $sessionID = $_SESSION["ddbms_session_id"];

        $ddbms = new DDBMSConnection($sessionID);
        if(!$ddbms->connected()) {
            $ddbms->connect("xxxx", "mysql", 3306,
                            true, true, true, true, true, true, true, true, true);
        }

        $res1ID = $ddbms->addResource("mysql",
                                      "slave1.net",
                                      3306,
                                      "db_name",
                                      "my_user",
                                      "my_user",
                                      "",
                                      "Grid Portal");
        $res2ID = $ddbms->addResource("mysql",
                                      "slave2.net",
                                      3306,
                                      "db_name",
                                      "my_user",
                                      "my_user",
                                      "",
                                      "MyProxy Server");

        $ddbms->execSQL("SHOW RESOURCES");
        while($row = $ddbms->fetchNextRow()) {
            for($i = 0; $i < count($row); ++$i) {
                printf("%s\t", $row[$i]);
            }
            printf("\n<br>\n");
        }
        printf("<br>\n");

        $ddbms->execSQL("CREATE DATABASE TestDB");
        $ddbms->execSQL("USE TestDB");

        $ddbms->execSQL("CREATE TABLE TestTB (ID INT, Name TEXT, Address TEXT) " .
                        "HFRAGMENT (ID IS ODD )(" . $res1ID . "),              " .
                        "          (ID IS EVEN)(" . $res2ID . ")              ");

        $ddbms->execSQL("INSERT INTO TestTB VALUES (0, 'Name 0', 'Address 0')");
        $ddbms->execSQL("INSERT INTO TestTB VALUES (1, 'Name 1', 'Address 1')");
        $ddbms->execSQL("INSERT INTO TestTB VALUES (2, 'Name 2', 'Address 2')");
        $ddbms->execSQL("INSERT INTO TestTB VALUES (3, 'Name 3', 'Address 3')");

        $ddbms->execSQL("SELECT * FROM TestTB");
        while($row = $ddbms->fetchNextRow()) {
            for($i = 0; $i < count($row); ++$i) {
                printf("%s\t", $row[$i]);
            }
            printf("\n<br>\n");
        }
        printf("<br>\n");

        $ddbms->execSQL("DROP DATABASE TestDB");

        $ddbms->removeResource($res1ID);
        $ddbms->removeResource($res2ID);

        $ddbms = null;
        ddbmsConnectionDestroySessionObject($sessionID);
    }
    catch (Exception $e) {
        echo nl2br(str_replace(" ", " ", $e->getMessage())) . "\n";
    }
?>
```

Please refer to the PHP specific documentation for the API.

7. Building Client Application using Network-Library API

This kind of application can be deployed in any computer which can connect to the VDDMS master node.

Below is an example C++ application:

```
#include <iostream>
using namespace std;

#include <DDBMSClient.h>
using namespace gclddbms_client;

int main()
{
    DBRow       dbr;
    DDBMSClient ddbms("master.net", 25OO1, "owner", "magic", true, true);

    const string& res1ID = ddbms.addResource("mysql",
                                             "slave1.net",
                                             33O6,
                                             "db_name",
                                             "my_user",
                                             "my_user",
                                             "",
                                             "Grid Portal");
    const string& res2ID = ddbms.addResource("mysql",
                                             "slave2.net",
                                             33O6,
                                             "db_name",
                                             "my_user",
                                             "my_user",
                                             "",
                                             "MyProxy Server");

    ddbms.execSQL("SHOW RESOURCES");
    while(ddbms.fetchNextRow(dbr)) {
        for(size_t i = O; i < dbr.size(); ++i) cerr << dbr[i] << "\t";
        cerr << endl;
    }
    cerr << endl;

    ddbms.execSQL("CREATE DATABASE TestDB");
    ddbms.execSQL("USE TestDB");

    ddbms.execSQL("CREATE TABLE TestTB (ID INT, Name TEXT, Address TEXT) "
                  "HFRAGMENT (ID IS ODD )(" + res1ID + "),               "
                  "          (ID IS EVEN)(" + res2ID + ")               ");

    ddbms.execSQL("INSERT INTO TestTB VALUES (O, 'Name O', 'Address O')");
    ddbms.execSQL("INSERT INTO TestTB VALUES (1, 'Name 1', 'Address 1')");
    ddbms.execSQL("INSERT INTO TestTB VALUES (2, 'Name 2', 'Address 2')");
    ddbms.execSQL("INSERT INTO TestTB VALUES (3, 'Name 3', 'Address 3')");

    ddbms.execSQL("SELECT * FROM TestTB");
    while(ddbms.fetchNextRow(dbr)) {
        for(size_t i = O; i < dbr.size(); ++i) cerr << dbr[i] << "\t";
        cerr << endl;
    }
    cerr << endl;

    ddbms.execSQL("DROP DATABASE TestDB");

    ddbms.removeResource(res1ID);
    ddbms.removeResource(res2ID);
}
```

Please refer to the C++ specific documentation for the API.

Below is an example Java application:

```
public class VddTest
{

    public static void main(String[] args)
    {
        DDBMSClient cl;

        try
        {
            cl = new DDBMSClient("192.168.159.128", 25001, "owner", "magic", true, true, 0);

            String res_id1 = cl.addResource("mysql", "localhost", 3306, "root", "123",
                                            "sarmady_linux");

            cl.execSQL("CREATE DATABASE TestDB1");
            cl.execSQL("USE TestDB1");
            cl.execSQL("CREATE TABLE User1 (ID INT NOT NULL UNIQUE PRIMARY KEY, " +
                       "Name TINYTEXT, Address TEXT) REPLICATE ('" + res_id1 + "')");

            System.out.println("Insert a Record Using Normal SQL and Show the Results :");
            cl.execSQL("INSERT INTO TestDB1.User1 VALUES('10','siamak', 'n.park')");

            // Show all records in Table
            cl.execSQL("SELECT * FROM TestDB1.User1");

            DBRow temp;
            while ((temp = cl.fetchNextRow(10)) != null)
            {
                for (int j = 0; j < temp.size(); j++)
                    System.out.print(temp.getValue(j) + " ");
                System.out.println();
            }

            System.out.println("\nInsert a Record Using Prepared Statement and " +
                               "Show the Results :");

            cl.prepSQL("INSERT INTO TestDB1.User1 VALUES(?a,?b, ?c)");
            cl.prepSetVal('a', "11");
            cl.prepSetVal('b', "'babak'");
            cl.prepSetVal('c', "'cyberjaya'");
            cl.prepExec();

            // Show all records in Table
            cl.execSQL("SELECT * FROM TestDB1.User1");

            while ((temp = cl.fetchNextRow(10)) != null)
            {
                for (int j = 0; j < temp.size(); j++)
                    System.out.print(temp.getValue(j) + " ");
                System.out.println();
            }


            System.out.println("Finished");

            cl.ping();
            cl.execSQL("DROP database TestDB1");
            cl.removeRescource(res_id1);
            cl.close();
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

Please refer to the Java specific documentation for the API.

Below is an example .NET application:

```
using System;
using System.Collections.Generic;
using System.Text;

using gclddbmsclient;

namespace gclddbmsclientTest
{
    class Program
    {
        static void dumpRow(ref DDBMSClient cDDBMSClient)
        {
            DBRow cDBRow = new DBRow();

            while (cDDBMSClient.fetchNextRow(ref cDBRow, 3))
            {
                for (uint i = 0; i < cDBRow.size(); ++i)
                {
                    Console.Write(cDBRow[i] + "\t");
                }
                Console.WriteLine();
            }
            Console.WriteLine();
        }

        static void Main(string[] args)
        {
            String username = "imran";
            String password = "imran";

            //debugging process
            Program program = new Program();

            // Main function for the VDDBMS
            // Initialize conenction
            DDBMSClient cDDBMSClient = new DDBMSClient("10.207.206.143",
                                                       25001,
                                                       username,
                                                       password,
                                                       false,
                                                       false,
                                                       10);

            Byte major = 0, minor = 0, revision = 0, bugfix = 0;

            // Request for server version
            cDDBMSClient.serverProtocolVersion(ref major, ref minor, ref revision,
                                               ref bugfix);
            Console.WriteLine("Server Version: " + (int)major + "." + (int)minor + "." +
                              (int)revision + "." + (int)bugfix);

            // Request for client version
            cDDBMSClient.clientProtocolVersion(ref major, ref minor, ref revision,
                                               ref bugfix);
            Console.WriteLine("Client Version: " + (int)major + "." + (int)minor + "." +
                              (int)revision + "." + (int)bugfix);

            // Ping the server
            Console.WriteLine("Ping: " + cDDBMSClient.ping());

            // Add resources
            String type = "mysql";
            String host = "localhost";
            UInt32 port = 3306;
            String desc = "Test Server";
            String resID1;
            resID1 = cDDBMSClient.addResource(type, host, port, "root", "gclddbms", desc);

            // Test SQL
            cDDBMSClient.evalSQL("CREATE TABLE TestDB.User1 (ID INT NOT NULL UNIQUE " +
                                 "PRIMARY KEY, Name TINYTEXT, Address TEXT)");
            Program.dumpRow(ref cDDBMSClient);

            cDDBMSClient.execSQL("CREATE DATABASE TestDB");
            cDDBMSClient.execSQL("USE TestDB");
```

```
            cDDBMSClient.execSQL("CREATE TABLE User1 (ID INT NOT NULL UNIQUE " +
                                 "PRIMARY KEY, Name TINYTEXT, Address TEXT) " +
                                 "REPLICATE(" + resID1 + ")");

            cDDBMSClient.execSQL("SHOW RESOURCES");
            Program.dumpRow(ref cDDBMSClient);

            cDDBMSClient.execSQL("SHOW DATABASES");
            Program.dumpRow(ref cDDBMSClient);

            cDDBMSClient.execSQL("SHOW TABLES");
            Program.dumpRow(ref cDDBMSClient);

            for (int iCounter = 0; iCounter < 10; iCounter++)
            {
                cDDBMSClient.execSQL("INSERT INTO User1 VALUES (" + iCounter +
                                     ", 'This is the name of person #" + iCounter +
                                     "', 'This is the address of person #" +
                                     iCounter + "')");
            }

            cDDBMSClient.execSQL("SELECT * FROM User1");
            Program.dumpRow(ref cDDBMSClient);

            // Just testing
            cDDBMSClient.discardRow(1);
            cDDBMSClient.discardRow(10);
            cDDBMSClient.discardAll();

            // Remove resources
            cDDBMSClient.execSQL("DROP DATABASE TestDB");
            cDDBMSClient.removeResource(resID1);

            Console.WriteLine("finished");
            Console.ReadKey();
        }
    }
}
```

Please refer to the .NET specific documentation for the API.