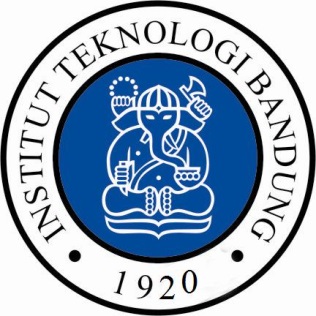
**Model Kluster Server**

**Tugas Besar IF3230**

**Semester II Tahun 2013 / 2014**

**Sistem Paralel dan Terdistribusi**

****

**Sabertooth Ganesha 2/3**

Muhamad Ihsan / 13511049

Habibie Faried / 13511069

Iskandar Setiadi / 13511073

Sekolah Teknik Elektro dan Informatika (STEI ITB)

Institut Teknologi Bandung

Jl. Ganesha No. 10, Bandung 40132

Tahun 2014

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Sekolah Teknik Elektro dan Informatika ITB** | - | Halaman |
| *IF3230-TB* | *9* |
| *11 Mei 2014* |

**Tahap I**

Pada tahap pertama ini, kami menggunakan bahasa Java untuk melakukan implementasi *client-server*.

|  |  |  |
| --- | --- | --- |
| Main | Tester.java | Kelas utama yang berfungsi sebagai Main dari program |
| Protokol & Struktur data | Client.java | Kelas yang berfungsi untuk menangani hal-hal yang terkait dengan *client* |
| Protocol.java | Kelas *socket* untuk transfer data dan *error handling* |
| Server.java | Kelas yang berfungsi untuk menangani hal-hal yang terkait dengan *server* |
| Struktur.java | Kelas yang berisikan struktur data yang digunakan |

Untuk struktur data, kami menggunakan **Hashtable<String, ArrayList<Data>>**, dimana String merupakan *key* yang merupakan nama *table* dan ArrayList<Data> merupakan *value*.

Data sendiri berisikan attribut berikut:

**public** String Key;

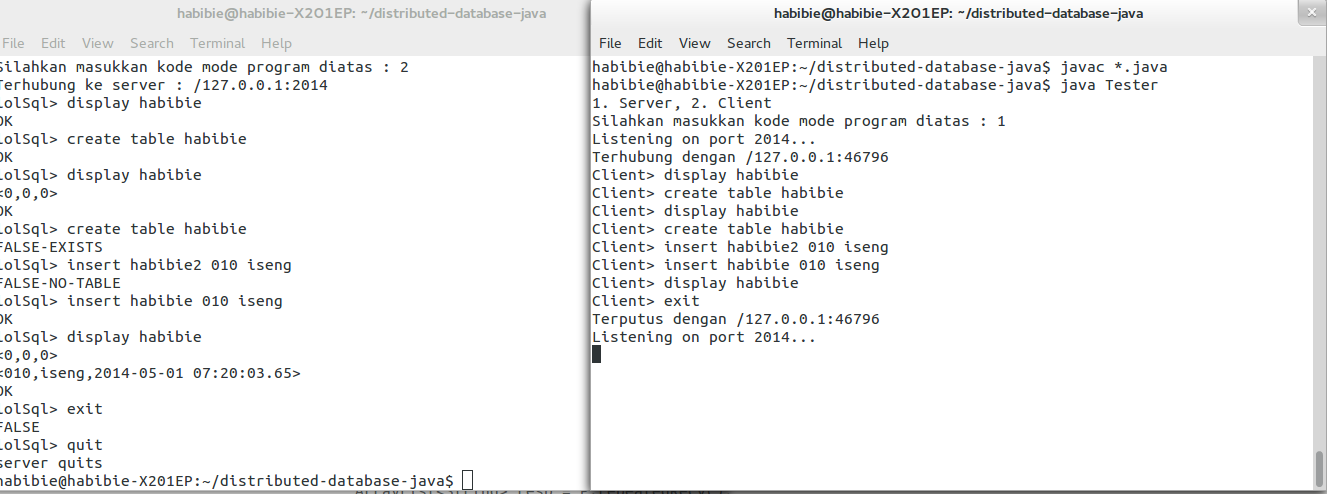
**public** String Value;

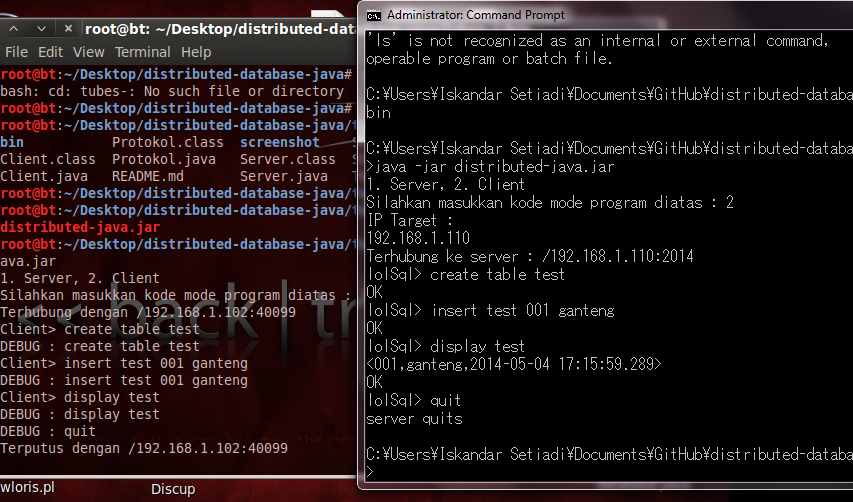
**public** String TimeStamp;

**public** **boolean** isPrint; //jika true maka akan ditampilkan, jika false maka tidak akan di print

Setiap *table* memiliki sekumpulan data yang terdiri dari *pair* <*Key, Value*> dengan *timestamp* adalah waktu saat data tersebut dimasukkan (waktu server). *isPrint* digunakan untuk menentukan *timestamp* terakhir yang akan ditampilkan.

Berikut ini adalah *screenshot* dari hasil *run* aplikasi:





Pembagian tugas:

1. Muhamad Ihsan : Mengerjakan Protocol.java
2. Habibie Faried : Melakukan rancangan terhadap Struktur.java
3. Iskandar Setiadi : Melakukan implementasi komunikasi Client.java dengan Server.java

**Tahap II**

Pada tahap kedua ini, kami mengembangkan prototip tahap I untuk membuat server terdistribusi. Berikut ini adalah *file* hasil implementasi yang kami lakukan:

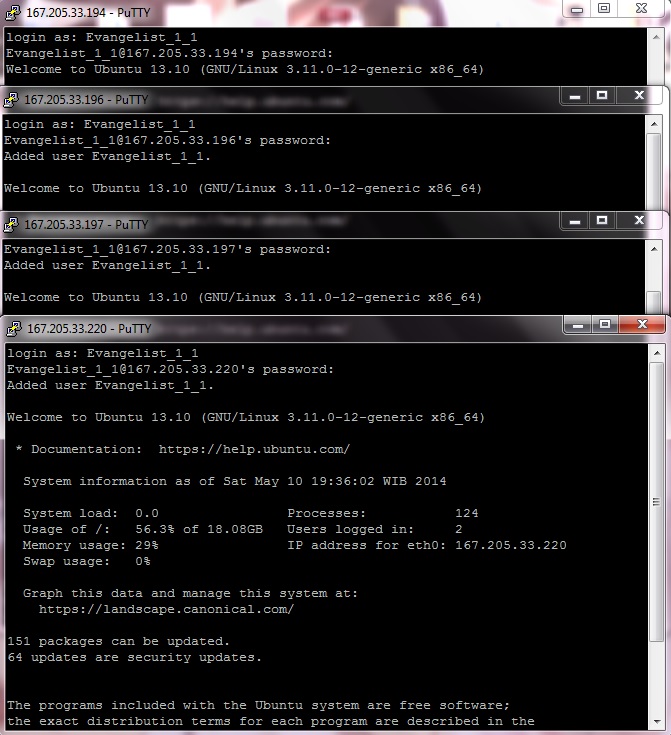
|  |  |  |
| --- | --- | --- |
| Main | Tester.java | Kelas utama yang berfungsi sebagai Main dari program |
| Protokol & Struktur data | Client.java | Kelas yang berfungsi untuk menangani hal-hal yang terkait dengan *client* |
| Protocol.java | Kelas *socket* untuk transfer data dan *error handling* |
| ListServer.java | Kelas yang berfungsi untuk menangani hal-hal yang terkait dengan *server* (dalam sistem terdistribusi) |
| Struktur.java | Kelas yang berisikan struktur data yang digunakan |
| *Hadoop-like* | trackers.java | Kelas yang merepresentasikan *namenode* |
| trackerc.java | Kelas yang merepresentasikan *datanode* |
| UnitTest | TestingUnit.java | Kelas ini digunakan untuk mode *client* yang berfungsi sebagai testing unit |

Pada implementasi ini, kami mengasumsikan bahwa jumlah *key* yang mungkin adalah **2048 (0 – 2047)** untuk mempermudah proses *debug* dan *checking*. Selain itu, kami menggunakan konsep *Hadoop* (HDFS) dalam implementasi ini. Pada prinsipnya, server terdiri atas dua bagian, yaitu satu *namenode* dan beberapa *datanodes* (tempat penyimpanan data). *Client* dapat berkomunikasi dengan *datanode* manapun, dan setiap *request* dari *client* akan diteruskan ke *namenode* untuk menentukan lokasi penyimpanan *table* dengan pasangan <*data*, *key*> yang dicari. Gambar[[1]](#footnote-1) berikut ini menunjukkan rancangan dari arsitektur HDFS yang kami terapkan:



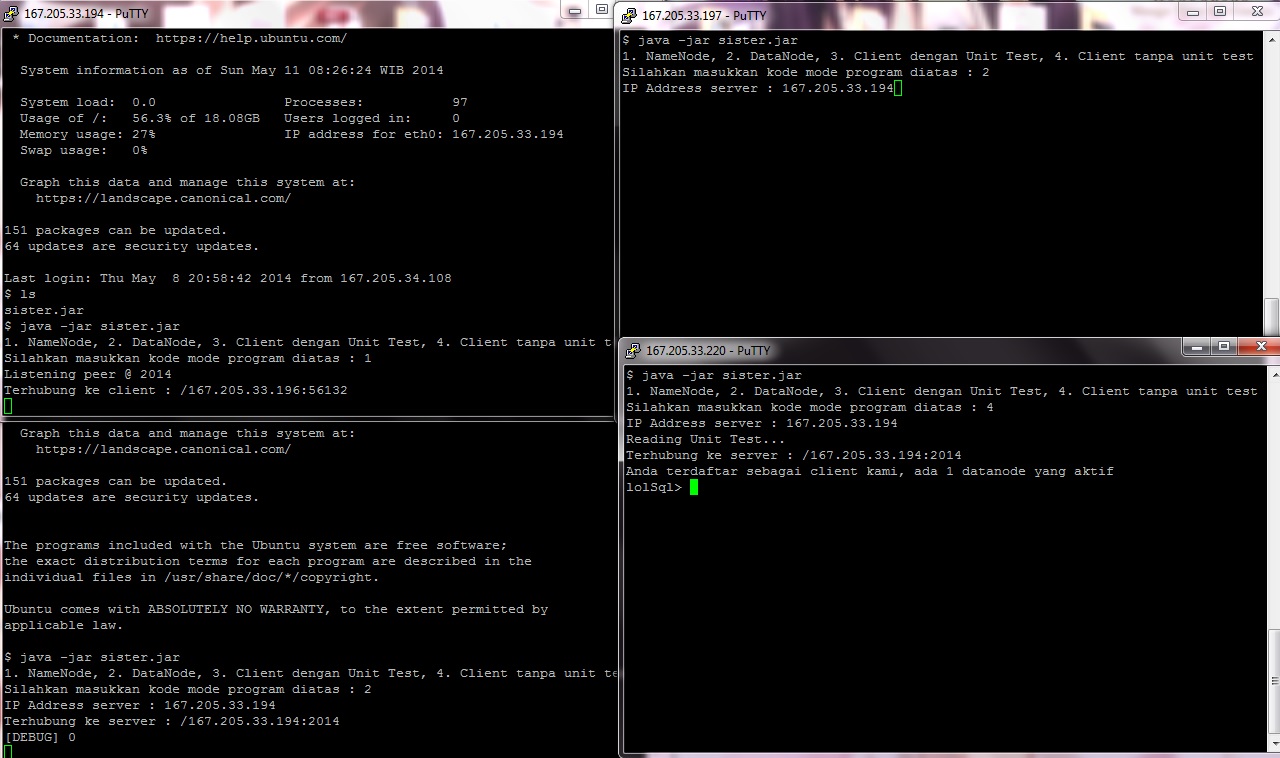
Untuk memulai simulasi ini, kami melakukan pemindahan data *sister.jar* dan *test.txt* ke alamat-alamat berikut ini (menggunakan *User* Evangelist\_1\_1):

1. Arctic – 167.205.33.194 🡪 *namenode*
2. Deux – 167.205.33.196 🡪 *datanode 1*
3. Tremors – 167.205.33.197 🡪 *datanode 2*
4. Delta – 167.205.33.220 🡪 *client*

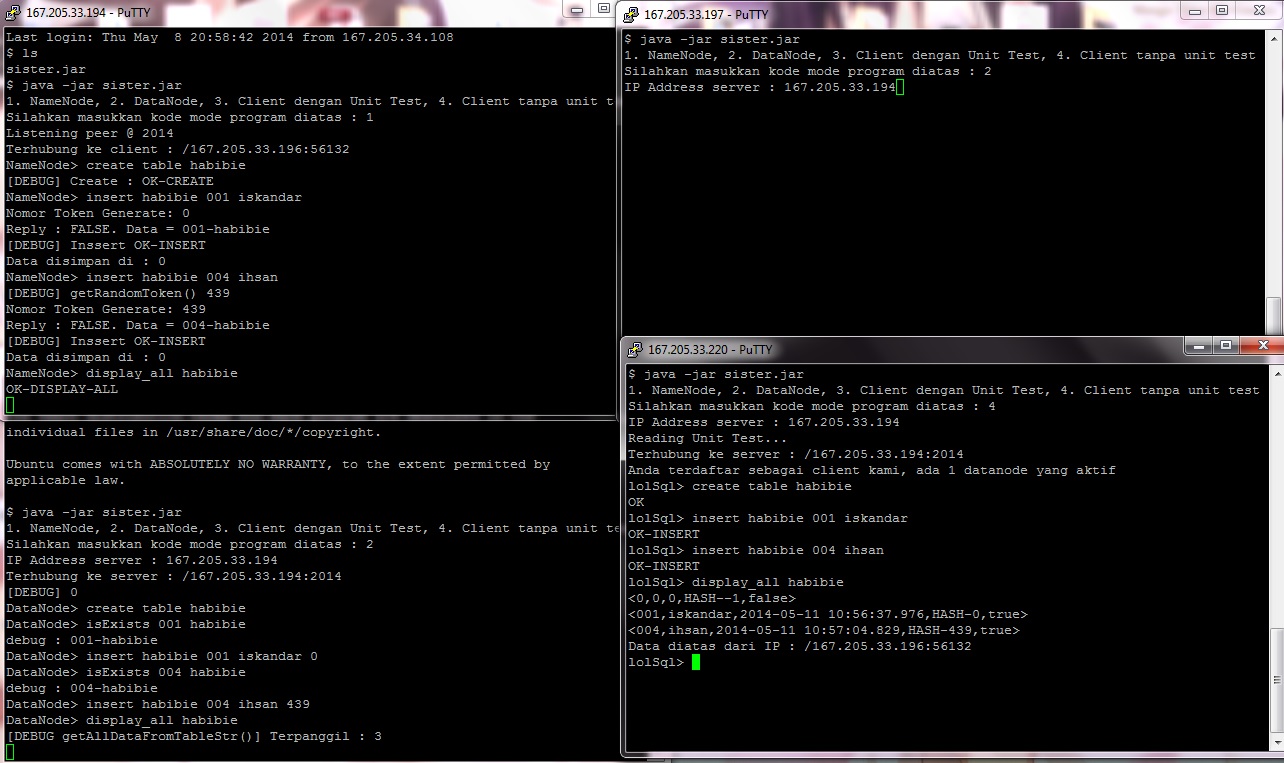


Kami menggunakan asumsi *mark <Data, Key>* dengan nilai <0, 0>.

Setelah itu, kami meng-*assign* masing-masing (kecuali *datanode 2* - 167.205.33.197) alamat sesuai dengan *jobdesk* masing-masing:

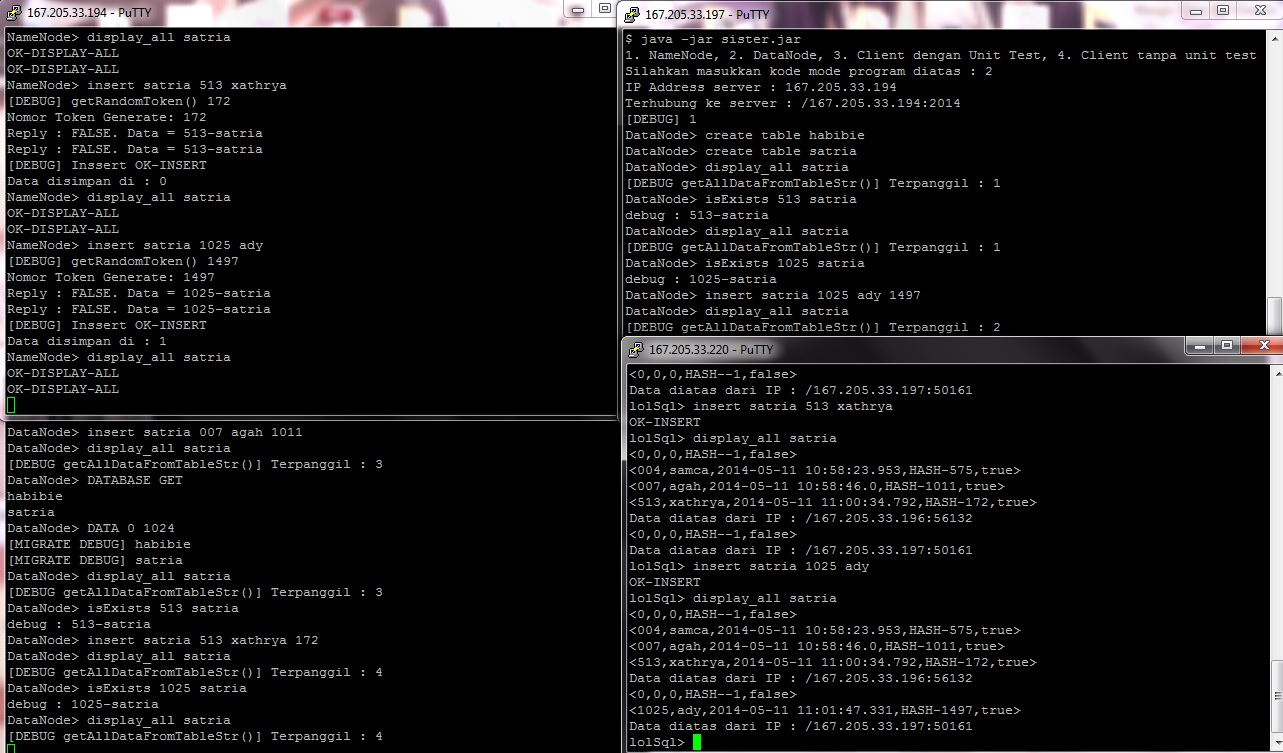


Selanjutnya, kita akan membuat tabel *habibie* dengan “create table habibie” serta melakukan insert *<001, iskandar>* dan *<004, ihsan>* kedalam tabel tersebut. Pada awalnya, semua data akan tersimpan di *datanode 1*.



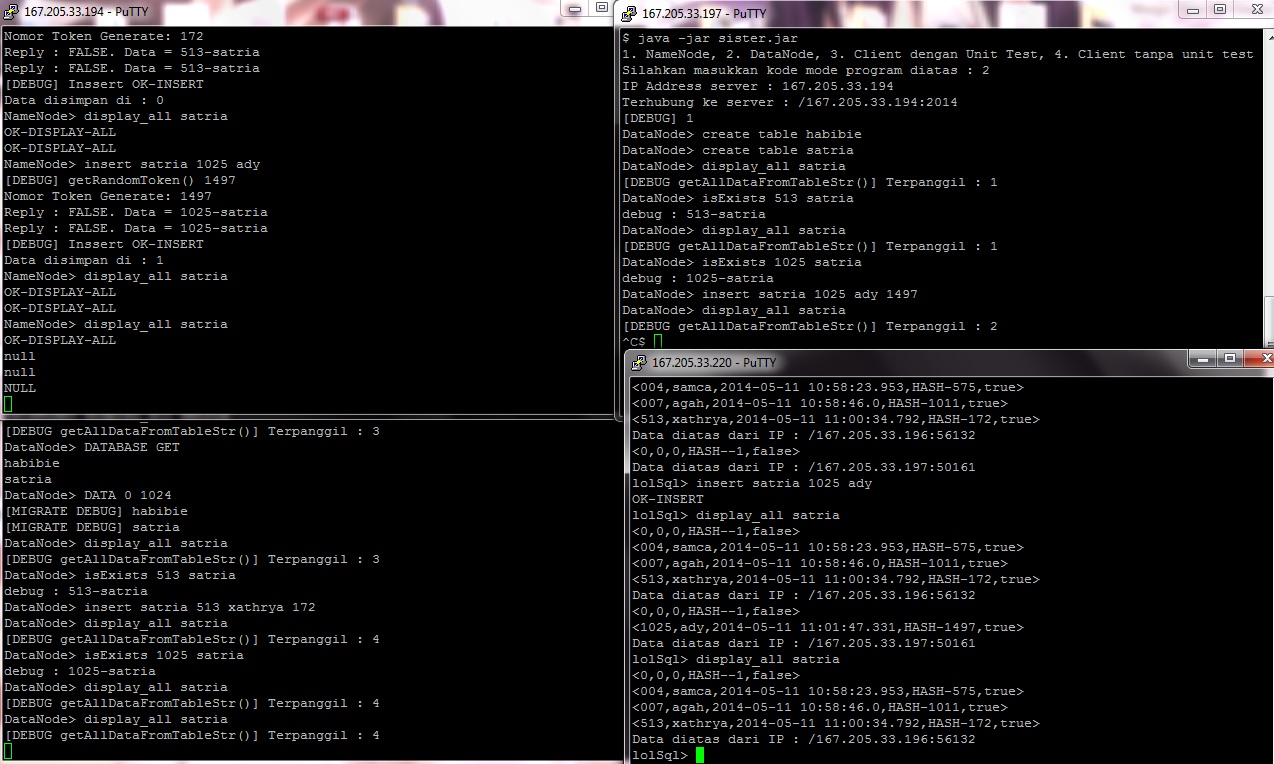
Pada tahap selanjutnya, kami menghubungkan *datanode 2* dengan *server* yang ada saat ini. Disini, semua *token* 0-1023 akan tersimpan di *datanode 1* sedangkan *token* 1024 – 2047 akan tersimpan di *datanode* *2*. Kami membuat tabel *Satria* dengan isi berikut:

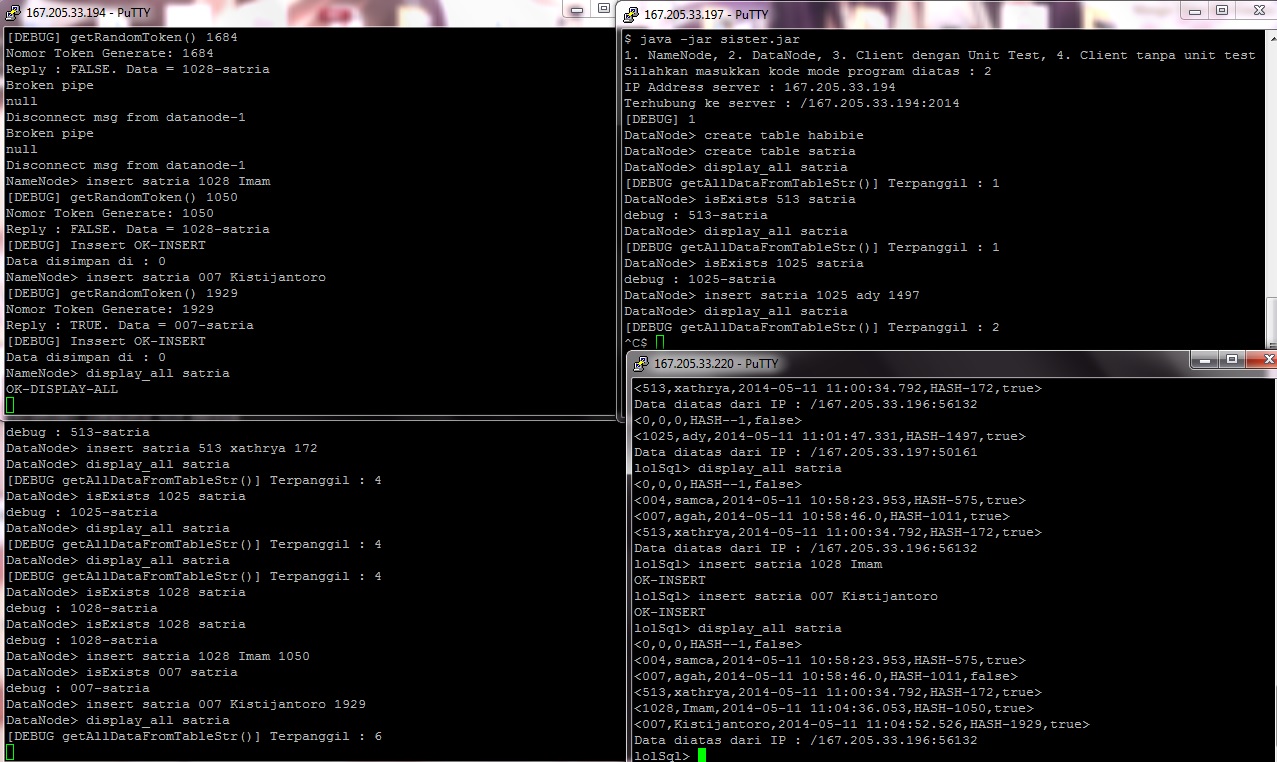
|  |  |  |
| --- | --- | --- |
| *Key* | *Data* | Tempat Penyimpanan |
| 004 | samca | *Datanode 1* |
| 007 | agah | *Datanode 1* |
| 513 | xathrya | *Datanode 1* |
| 1025 | ady | *Datanode 2* |



Untuk *error handling*, kami mengasumsikan bahwa semua alokasi ke *datanode* yang masih aktif. Alokasi ini dilakukan oleh *namenode*, sehingga pada simulasi ini, kami men-*shutdown* *datanode 2* secara paksa, sehingga semua alokasi dikembalikan ke *datanode* *1*.

Kami menambahkan tupel *<1028, Imam>* dan *<007, Kistijantoro>* ke *server* yang masih aktif saat ini (tabel *satria*). Tupel *<1028, Imam>* digunakan untuk menunjukkan bahwa semua alokasi dikembalikan ke *datanode 1*, sedangkan tupel *<007, Kistijantoro>* digunakan untuk menunjukkan *timestamp* yang hanya menampilkan data terakhir, namun menyimpan semua *history* dari data yang pernah masuk.





1. Sumber : <http://hadoop.apache.org/docs/r1.2.1/images/hdfsarchitecture.gif> <diakses 11 Mei 2014> [↑](#footnote-ref-1)