**Inhalt**

**1 Aufgabenstellung Seite 2**

**2 Grobstruktur der Miniwelt Seite 2**

**3 Feinstruktur Der Miniwelt Seite 3**

3.1 Die Proxmox Weboberfläche Seite 3

3.2 Die Entitäten Seite 4

3.3 Attribute der Entitäten Seite 5

3.4 Beziehungen der Entitäten Seite 6

**4 Entity Relationship Modell Seite 8**

**5 Realationenmodell Seite 9**

**6 Ordner und Dateistruktur Seite 10**

**1 Aufgabenstellung**

Entwicklung einer Datenbank für ein Proxmoxcluster, um die unzähligen darauf liegenden virtuellen Maschinen tabellarisch zu erfassen, und zu verwalten.

**2 Grobstruktur der Miniwelt**

Der "Proxmox Python SQL config extractor to Database injector" wurde von zwei IHK-Umschülern und einem Auszubildenden zum Fachinformatiker für Systemadministration ins Leben gerufen.

Zunächst stand der Spaß im Vordergrund, Dienste für die Klasse zu hosten. Dann kamen die ersten Dienste für Private und familiäre Anwender hinzu.

Da der Verwaltungsaufwand der schier unendlichen Menge Virtueller Maschinen uns bereits nach Sekunden überforderte, entstand die Idee, einen Teil der Systemadministration zu automatisieren.

Die Virtuellen Maschinen werden zwar nach einem Schema erstellt, welches jedoch gewisse Freiheiten erlaubt.

So variieren z. B. die Menge der CPU Kerne, der RAM, das OS und vieles mehr, es muss dadurch auf die Verfügbaren Recourcen geachtet werden. Außerdem müssen manche Dinge, so zum Beispiel Ports, einmalig sein.

Aus den Konfigurationsdateien des Hostsystems sollen die daten nun ausgelesen werden und die Datensätze zur weiteren Verarbeitung und Aufbewahrung in eine Oracle Datenbank übertragen werden. Hierzu wird ein Python Skript, das Dateien auslesen, und mit SQL befehlen daten in eine Datenbank schreiben kann Entwickelt.

Der "Proxmox Python SQL config extractor to Database injector" benötig eine Datenbank zur Verwaltung folgender Punkte:

-> Erfassung aller Virtuellen Maschinen und Ressources innerhalb von Proxmox Clustern!

-> Zuteilung entsprechender Ressourcen

In weiteren Ausbaustufen ist geplant

-> Reservierung von Hardware für künftige Projekte

-> Visualisierung der Ressourcen mittels Graphen in Python

-> Graphische Ausgabe der Virtuellen Netzwerk Infrastruktur

**3 Feinstruktur Der Miniwelt**

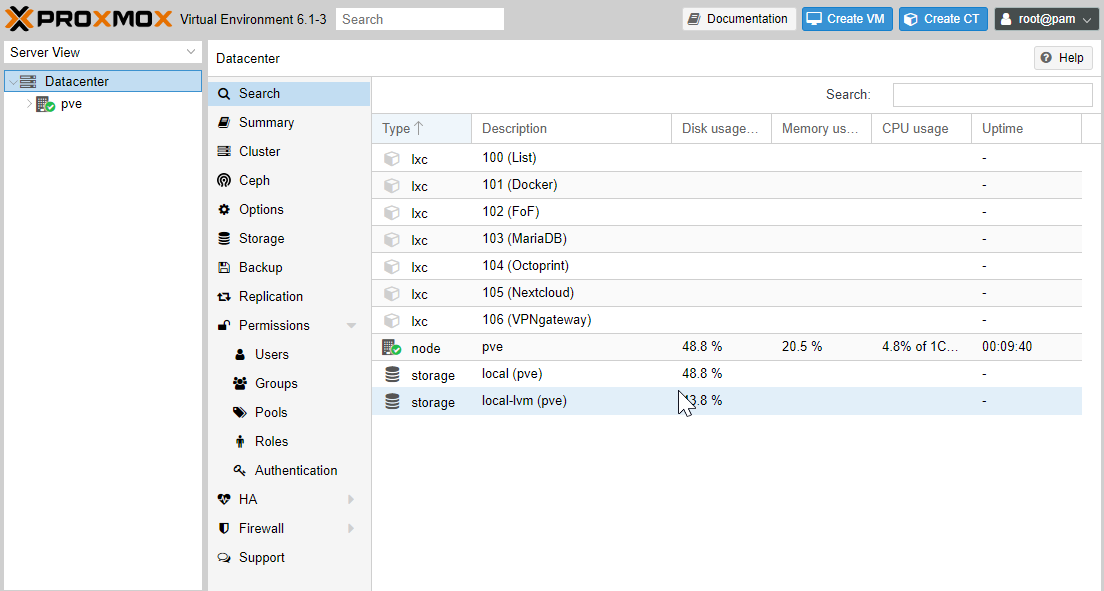
Eine virtuelle Maschine bezeichnet in der Informatik eine Software seigte Lösung zur Abbildung eines Computers, mit all seinen Bestandteilen, in einer rein virtuellen Umgebung.

Der Host Ist dabei meist ein Betriebssystem bzw. eine Software die speziell dafür designt ist virtuelle Hardware bereitzustellen und die Einrichtung der Betriebssysteme zu bewerkstelligen.

Es gibt verschiedene Lösungen zur Virtualisierung in unserem Fall fiel die Wahl auf Proxmox Virtual Environment (PVE), eine auf Debian basierende Open-Source Virtualisierungsplattform. Es ist kosten frei nutzbar, auch für kommerzielle Zwecke, es werden lediglich Support Angebote von Proxmox vertrieben.

Der open-source Aspekt und die Tatsache das das System sich relativ umfangreich über eine web Oberfläche bedienen und konfigurieren lässt sind die Hauptgründe für die Wahl des Systems.

3.1 Die Proxmox Weboberfläche



Abbildungx.1: Overview der auf dem Host installierten VM-Clients und Festplatten.

3.2 Die Entitäten

Es gibt zu diesem Zeitpunkt 3 Entitäten:

1. Host
2. Client
3. Subnet

Ein Host kann mehrere Client Systeme verwalten und besitzt eventuell mehrere Festplatten (HDDs).

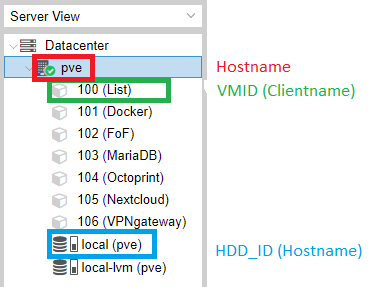


Abb.X.2 (Screenshot Proxmox Web-Oberfläche)

Das Subnet ist der Netzwerkbereich in dem sich die IP-Adresse des Hosts oder des Client befinden.

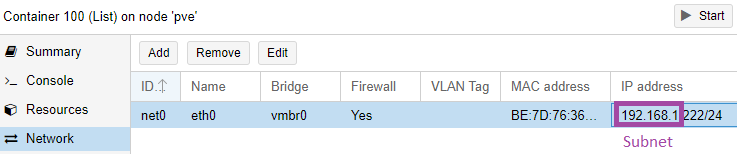


Abb. X.3 (Screenshot Client Netzwerk Configuration)

3.3 Attribute der Entitäten

In den Vorherigen Abbildungen sind einige Daten (Attribute) die später in die Datenbank überführt werden zu sehen, im Folgenden werden tabellarisch ihre, für Datenverarbeitungszwecke relevanten, Eigenschaften aufgeführt .

3.3.1 Host

|  |  |
| --- | --- |
| **Hostname** | Eindeutiger Name, vom admin oder vom Hostsystem vergeben |
| Datentyp | Varchar2 |
| Wertevergabe | Vergeben durch Admin im Proxmox interface aderseits automatisch |
| Default-Wert | Nein |
| Null-Wert erlaubt | Nein |
| Duplikate Erlaubt | Nein |
| PK | Ja |
| Beispiel werte | pve, Horst, Hostinator,… |

|  |  |
| --- | --- |
| **hIP** | Eindeutige Netzwerk-Adresse |
| Datentyp | Varchar2 |
| Wertevergabe | Durch Admin (Statisch) DHCP-Server (Dynamisch) (Router, VPN, etc) |
| Default-Wert | Nein |
| Null-Wert erlaubt | ja |
| Duplikate Erlaubt | Nein |
| PK | Ja |
| Beispiel werte | pve, Horst, Hostinator,… |

|  |  |
| --- | --- |
| HDDs | Mehrwertiges Attribut |
| **HDD\_ID** | Edeutige Bezeichnung der Festplatte | |
| Datentyp | Varchar2 |
| Wertevergabe | Durch Admin (Statisch) DHCP-Server (Dynamisch) (Router, VPN, etc) |
| Default-Wert | Nein |
| Null-Wert erlaubt | ja |
| Duplikate Erlaubt | Nein |
| PK | Ja zusammen mit Hostname |
| Beispiel werte | pve, Horst, Hostinator,… |
| **Hostname** | Fremdschlüssel aus Host | |
| Datentyp | Varchar2 |
| Wertevergabe | Vergeben durch Admin im Proxmox interface oder automatisch |
| Default-Wert | Nein |
| Null-Wert erlaubt | Nein |
| Duplikate Erlaubt | ja |
| PK | Ja zusammen mit HDD\_ID |
| Beispiel werte | pve, Horst, Hostinator,… |

3.3.2 Client

|  |  |
| --- | --- |
| **VMID** | Eindeutige 3stellige ID, vom Hostsytem vergeben |
| Datentyp | Number |
| Wertevergabe | Vergeben durch Admin im Proxmox interface |
| Default-Wert | Nein |
| Null-Wert erlaubt | Nein |
| Duplikate Erlaubt | Nein |
| PK | Ja |
| Beispiel werte | 111, 222, 333,… |

|  |  |
| --- | --- |
| **Clientname** | Entspricht dem Hostnamen des jeweiligem Clientsystems |
| Datentyp | Varchar2 |
| Wertevergabe | Bei Einrichtung der VM durch Admin |
| Default-Wert | Nein |
| Null-Wert erlaubt |  |
| Duplikate Erlaubt | Nein |
| PK | Nein |
| Beispiel werte | debianOraXE,Win10test |

|  |  |
| --- | --- |
| **cIP** | Eindeutige Netzwerk-Adresse |
| Datentyp | Varchar2 |
| Wertevergabe | Durch Admin (Statisch) DHCP-Server (Dynamisch) (Router, VPN, etc) |
| Default-Wert | Nein |
| Null-Wert erlaubt | ja |
| Duplikate Erlaubt | Nein |
| PK | Nein |
| Beispiel werte | 192.168.9.254, 123.10.10.100,… |

|  |  |
| --- | --- |
| **Disk\_Space** | Entspricht der größe des virtuelen laufwerks der VM |
| Datentyp | Varchar2 |
| Wertevergabe | Vergebn durch admin im Proxmox interface |
| Default-Wert | Nein |
| Null-Wert erlaubt | Nein |
| Duplikate Erlaubt | Ja |
| PK | Nein |
| Beispiel werte | 20G, 100G, 8G, …. |

|  |  |
| --- | --- |
| **OS\_Type** | Ist die art/ der name des Betriebssystems |
| Datentyp | Varchar2 |
| Wertevergabe | Durchauslesen der configuratins datei |
| Default-Wert | Nein |
| Null-Wert erlaubt |  |
| Duplikate Erlaubt | Ja |
| PK | Nein |
| Beispiel werte | Debian, ubuntu, win10, …. |

|  |  |
| --- | --- |
| **Cores** | Anzahl der virtuelen prozessor kerne |
| Datentyp | Number |
| Wertevergabe | Durch admin bei einrichtung der VM |
| Default-Wert | Nein |
| Null-Wert erlaubt | Nein |
| Duplikate Erlaubt | Ja |
| PK | Nein |
| Beispiel werte | 1, 3, 8, 16, … |

|  |  |
| --- | --- |
| **RAM** | VIrueller arbeitsspeicher in gb |
| Datentyp | Varchar2 |
| Wertevergabe | Durch admin bei einrichtung der VM |
| Default-Wert | Nein |
| Null-Wert erlaubt | Nein |
| Duplikate Erlaubt | Ja |
| PK | Nein |
| Beispiel werte | 1024 , 8000, 1600, … |

3.3.3 Subnet

|  |  |
| --- | --- |
| **Subnet\_ID** | Virtueller Arbeitsspeicher in GB |
| Datentyp | Varchar2 |
| Wertevergabe | Durch FISI bei Einrichtung der Netzwerkumgebung |
| Default-Wert | Nein |
| Null-Wert erlaubt | Nein |
| Duplikate Erlaubt | Ja |
| PK | Ja |
| Beispiel werte | 192.168.1, 192.10.10, …. |

3.4 Beziehungen der Entitäten

3.4.1 Verwalten:

|  |  |  |
| --- | --- | --- |
|  | Beteiligung | Kardinalität |
| Host | 0  Jeder host braucht keinen Client | N  Ein Host kann mehrere Clients haben |
| Client | 1  Jeder Client Braucht einen Host | 1  Ein Client kann nur einen Host haben |

3.4.2 Host Befindet sich in:

|  |  |  |
| --- | --- | --- |
|  | Beteiligung | Kardinalität |
| Host | 0  Ein host muss sich nicht Zwingend in einem Subnet befinden | N  Ein Host kann sich in mehreren Subnets befinden |
| Subnet | 0  Ein Subnet Braucht keinen Host | N  Ein Subnet kann mehrere Hosts haben |

3.4.3 Client Befindet sich in:

|  |  |  |
| --- | --- | --- |
|  | Beteiligung | Kardinalität |
| Client | 0  Ein Client muss sich nicht Zwingend in einem Subnet befinden | N  Ein Host kann sich in mehreren Subnets befinden |
| Subnet | 0  Ein Subnet Braucht keinen Client | N  Ein Subnet kann mehrere Clients haben |

**4 Entity Relation Modell**

4.1 Gesamt Darstellung



Abb.4.1 (Gesamt Darstellung des Entity Relationship-Modells.)

4.2 Detail- Darstellung

Client



Abb.4.2 (Entität Client)

Subnet Host

Abb.4.3(Entität Subnet) Abb.4.4(Entität Client)

**5 Realationenmodell**

Host (hIP , Hostname)

HDDs (HDD\_ID , Hostname )

Client (VMID , OS\_Type , RAM , Cores , disk\_space , cIP , Clientname, Hostname )

Subnet (SubnetID)

hostSubnet (SubnetID, Hostname) FK hostsubnet.Hostname 🡪 PK Host.Hostname

clientSubnet (SubnetID ,VMID) FK clientsubnet.VMID 🡪 PK Client.VMID

**6 Ordner- und Dateistruktur**

Das Verwalten der Datenbank wird mit einem in SQL plus geschriebenen Menü übernommen.

Folgende Darstellungen zeigen das Haupt und ein Sub Menü.

Außerdem gibt es im Python Skript ein Menü welches es ermöglicht auf SQL\*Plus zu zugreifen.

